## *Optimisation of Bluetooth wireless personal area networks*

Robinson, Craig

# Optimisation of Bluetooth Wireless Personal Area Networks

Craig Robinson



Presented in partial fulfillment of the degree of

Doctor of Philosophy

Centre for Electronic Systems

School of Engineering

University of Durham

England

January 2004

# Optimisation of Bluetooth Wireless Personal Area Networks

**Craig Robinson, MEng**

## Abstract

In recent years there has been a marked growth in the use of wireless cellular telephones, PCs and the Internet. This proliferation of information technology has hastened the advent of wireless networks which aim to increase the accessability and reach of communications devices. Ambient Intelligence (AmI) is a vision of the future of computing in which all kinds of everyday objects will contain intelligence. To be effective, AmI requires Ubiquitous Computing and Communication, the latter being enabled by wireless networking. The IEEE's 802.11 task group has developed a series of radio based replacements for the familiar wired ethernet LAN. At the same time another IEEE standards task group, 802.15, together with a number of industry consortia, has introduced a new level of wireless networking based upon short range, ad-hoc connections. Currently, the most significant of these new Wireless Personal Area Network (WPAN) standards is Bluetooth, one of the first of the enabling technologies of AmI to be commercially available.

Bluetooth operates in the internationally unlicensed Industrial, Scientific and Medical (ISM) band at 2.4 GHz. Unfortunately, this spectrum is particularly crowded. It is also used by: WiFi (IEEE 802.11); a new WPAN standard called ZigBee; many types of simple devices such as garage door openers; and is polluted by unintentional radiators. The success of a radio specification for ubiquitous wireless communications is, therefore, dependant upon a robust tolerance to high levels of electromagnetic noise. This thesis addresses the optimisation of low power WPANs in this context, with particular reference to the physical layer radio specification of the Bluetooth system.

Bluetooth modulates data onto a carrier frequency using a form of digital FM called Gaussian Frequency Shift Keying (GFSK). To provide resistance to interference the frequency of a Bluetooth carrier hops onto one of seventy nine 1 MHz wide channels, approximately once every 625 μs. The Bluetooth specification defines the tuning accuracy after each hop. However, the allowable error is an appreciable fraction of the magnitude of frequency change used to encode data. Simulation results, presented herein, show that the effect of this can be to significantly degrade a Bluetooth receiver's ability to correctly demodulate transmitted data in the presence of noise. The minimum signal to noise ratio (Eb/No) almost doubles to 30dB with a maximally mistuned transmitter. If the receiver is also maximally mistuned, the reception of a packet becomes impossible regardless of SNR. It is clear, therefore, that optimisation of this aspect of a Bluetooth radio is possible by removing the effect of carrier frequency errors upon the reception of data. A means of measuring the modulation characteristics of a Bluetooth radio is also desirable.

A method of extracting a Bluetooth transmitter's modulation characteristics was developed and tested in conjunction with Advantest in Munich. The implementation is software based and so is suited to general purpose automatic test equipment not requiring dedicated Bluetooth testers or communications analysers. Its efficacy was verified using an Advantest T7611 RF IC test system. In over two and half thousand tests the initial carrier frequency error of a Bluetooth signal was consistently estimated to within 900 Hz of the true value of up to 75 kHz error on a carrier frequency in the ISM band.

The demonstrable effect of carrier frequency errors on a Bluetooth receiver's performance is the stimulus for a receiver tolerant to such errors. The design of such a receiver is presented along with simulations of demonstrating its superior performance over non tolerant receivers. The full range of frequency errors is compensated for with a maximum performance degradation of 2.5 dB SNR.

In the final part of the thesis a prototype Bluetooth test instrument is described.

It was developed for use with the School of Engineering's Mixed Signal IC Test Station. Work in the previous chapters was addressed by incorporating the ability to manipulate the carrier frequency of signals to a device under test. The prototype also employs a novel software interface that combines Labview test management software with Bluetooth protocol stack software.

Inevitably, however, the development of WLANs and WPANs will result in increased noise levels within a limited spectrum. Therefore, it is important that each radio system uses bandwidth as efficiently as possible. This thesis highlights an aspect of the Bluetooth specification which can render it less than optimally efficient, offering mechanisms to quantify and compensate for this effect.

# Declaration

The work in this thesis is based on research carried out in the Centre for Electronic Systems, School of Engineering, University of Durham, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

# Acknowledgements

I would like to thank my supervisor, Professor Alan Purvis, for his support and guidance during the course of my studies which were generously supported by a studentship from Durham University's School of Engineering, for which I am very grateful.

I would also like to acknowledge the support of: Armin Lechner, Michael Hoy and Allan Jarret of Advantest Europe for allowing me to use their RF IC tester; Alan Oakley of Credence/IMS for his assistance with the MSTS; and my colleagues in Durham who are a pleasure to work alongside.

Finally, thank-you to my wife Celia. Without her encouragement, love and support I would never have even begun a PhD, never mind written this thesis.

# Contents

# List of Figures

# List of Tables

# Nomenclature

| Symbol or Abbreviation | Explanation |
|---|---|
| $\alpha$ | Carrier frequency drift ($Hzs^{-1}$) |
| $\theta$ | Phase |
| $\sigma$ | Variance |
| $\mu$ s | Micro-second |
| 802.11 | IEEE standard 802.11 (WiFi or Wireless LAN) |
| 802.15 | IEEE standard 802.15 (WPAN standards) |
| A | Amplitude (of a signal) |
| a | substitute for $\alpha$ in Figures |
| ACL | Asynchronous Connection-Less |
| ADC | Analogue to Digital Converter |
| AFH | Adaptive Frequency Hopping |
| AMI | Ambient Intelligence |
| AR | Auto-Regressive |
| ARMA | Auto-Regressive Moving Average |
| ASIC | Application Specific Integrated Circuit |
| ATE | Automated Test Equipment |
| AWG | Arbitrary Waveform Generator |
| AWGN | Additive White Gaussian Noise |
| B | Bandwidth (c.f. filter in GFSK) |
| BCSP | BlueCore Serial Protocol |
| BER | Bit Error Rate |
| BRAN | Broadband Radio Access Network |
| BT | Bandwidth Time product (GFSK) |
| CAN | Controller Area Network |
| CRC | Cyclic Redundancy Check |
| CRLB | Cramer-Rao Lower Bound |
| CSR | Cambridge Silicon Radio |
| CZT | Chirp Z Transform |
| DAC | Digital to Analogue Converter |

| Symbol or Abbreviation | Explanation |
|---|---|
| dB | Deci-bel |
| dBm | Deci-bel, power referred to 1 milli-watt |
| DC | Direct Current (0 Hz) |
| DFE | Decision Feedback Equaliser |
| DFT | Direct Fourier Transform |
| DSL | Digital Subscriber Line |
| DSP | Digital Signal Processing |
| DUT | Device Under Test |
| E | Bit energy |
| Eb/No | Bit energy to noise ratio |
| ETSI | European Telecommunication Standards Institute |
| $f_d$ | Peak frequency deviation (GFSK) |
| FEC | Forward Error Check |
| FFT | Fast Fourier Transform |
| FH | Frequency Hopping |
| $f_i$ | Instantaneous frequency |
| FIFO | First In First Out |
| FIR | Finite Impulse Response |
| fm | Frequency Modulation |
| $f_o$ | Frequency offset |
| $f_s$ | Sampling frequency |
| $f_c$ | Carrier frequency |
| GFSK | Gaussian Frequency Shift Keying |
| GHz | Giga-Hertz |
| GSM | Global System for Mobiles |
| GUI | Graphical User Interface |
| h | Modulation index (GFSK) |
| H(w) | Transfer function (of a filter) |
| HCI | Host Controller Interface |

| Symbol or Abbreviation | Explanation |
|---|---|
| HEC | Header Error Check |
| I | In-phase |
| IaD | Integrate and Dump |
| IC | Integrated Circuit |
| IEEE | Institute of Electrical and Electronic Engineers |
| IF | Intermediate Frequency |
| IIR | Infinite Impulse Response |
| IMS | Integrated Measurement Systems |
| IPC | Inter-Process Communication |
| IrDA | Infra-red Data Association |
| ISM | Industrial Scientific and Medical (frequency band) |
| ISTAG | Information Society Technology Advisory Group |
| ITRS | International Technology Roadmap for Semiconductors |
| JTAG | Joint Test Advisory Group |
| KHz | Kilo-hertz |
| L2CAP | Logical (L)ink Control and Adaptation Protocol |
| LAN/WLAN | Local Area Network/Wireless LAN |
| LAP | Lower Address Part |
| LM(P) | Link Manager (Protocol) |
| LO | Local Oscillator |
| LOS | Line Of Sight |
| LSI | Large Scale Integration |
| LTE | Linear transversal Equaliser |
| MA | Moving Average |
| MAC | Medium Access Control |
| MAN | Metropolitan Area Network |
| MCRLB | Modified Cramer-Rao Lower Bound |
| MHz | Mega-Hertz |
| MLE | Maximum Likelihood Estimator |

| Symbol or Abbreviation | Explanation |
|---|---|
| MLSE | Max Likelihood Sequence Estimator |
| MSTS | Mixed Signal Test System |
| MUSIC | Multiple Signal Classifiaction |
| NRZ | Non-Return to Zero |
| OBEX | Object Exchange (protocol) |
| OFDM | Orthogonal Frequency Division Multiplexed |
| PAN | Personal Area Network |
| PLL | Phase Lock Loop |
| ppm | parts per million |
| PRBS | Pseudo-Random Binary Sequence |
| PSD | Phase Shift Discriminator |
| Q | Quadrature |
| RF | Radio Frequency |
| RFComm | Bluetooth serial protocol emulation layer |
| Rx | Receive |
| SCO | Synchronous Connection Orientated |
| SDP | Service Discovery Protocol |
| SIG | Special Interest Group (Bluetooth SIG) |
| SNR | Signal to Noise Ratio |
| SoC | System on Chip |
| SSB-AM | Single Side Band Amplitude Modulation |
| s(t) | Message signal |
| T | Symbol duration |
| $T_w$ | Length of truncation window in fft (section 6.11 only) |
| TCS | Telephony Control protocol Specification |
| TDD | Time Division Duplex |
| Th | Decision Threshold |
| Tx | Transmit |
| UART | Universal Asynchronous Receiver Transmitter |

| Symbol or Abbreviation | Explanation |
|---|---|
| UMTS | Universal Mobile Telecommunications System |
| USB | Universal Serial Bus |
| UWB | Ultra Wide Band |
| VCO | Voltage Controlled Oscillator |
| VI | Virtual Instrument |
| VISA | Virtual Instrument Software Architecture |
| VNA | Vector Network Analyser |
| VXI | VME eXtensions for Instrumentation |
| Wifi | IEEE 802.11 |
| WPAN | Wireless Personal Area Network |
| $\overline{x}$ | Mean value (in Table 6.1) |
| VHDL | VHSIC Hardware Definition Language |

# Chapter 1

# Ubiquitous Communications

## 1.1 Introduction

The last decade has seen a proliferation of wireless communications devices. By the end of February 2003, 75% of adults in the United Kingdom, approximately 34.5 million people, used mobile phones [1]. This is an enormous number, especially when compared to the equivalent figure of 26% from only 4 years earlier. However, that this is unsurprising and, indeed, may seem lower than expected, speaks of the extent to which cellular radio has pervaded society. The ability to communicate with anyone, anywhere, at all times is no longer perceived as a luxury but as a normal part of many people's lives. This growth in radio communications has been fuelled by the improvements in digital and RF circuit fabrication, very large scale integration and miniturisation techniques which have made low power, low cost and reliable mobile radio devices feasible [2].

In parallel with the advance in wireless communications has been the development of the internet and World Wide Web. Between January and March 2003 an estimated 47% of households in the UK had access to the internet, with 17% using a broadband connection [3]. However, the vast majority of internet access (98% [3]) is attained through a PC, with its inherent lack of mobility and inconvenience of



location [4]. This is at odds with the expectation of mobile communications derived from cellular radio usage. Therefore, a natural progression is that access to the internet's information and entertainment services becomes mobile. This creates a present day demand for wireless, data-centric communications in homes, offices and elsewhere.

In the coming decades the need for wireless data connections will only increase. The European Community's Information Society Technology Advisory Group (ISTAG) has a vision of 'Ambient Intelligence' (AmI) in which:

> "People are surrounded by intelligent, intuitive interfaces that are embedded in all kinds of objects and an environment that is capable of recognising and responding to the presence of different individuals in a seamless, unobtrusive and often invisible way."[5]

AmI requires ubiquitous, or pervasive, computing, a concept first articulated by Mark Weiser at the XEROX Parc laboratory in 1988 [6]. In essence it means that technology recedes into the background, implying a need for ubiquitous communications in the form of ad-hoc networks between numerous, perhaps portable, very low cost computing devices [7]. The clear choice for forming such networks is with wireless connectivity [8].

Radio communications may also have a role to play in upholding Moore's Law. As the minimum feature size in integrated circuits shrinks to deep sub-micron levels, the dimensions of the metallic conductors are scaling proportionately. The resultant increases in parasitic resistance, capacitance and inductance are beginning to affect circuit performance to the point that they are a significant barrier to the development of Ultra LSI technology [9]. Of particular concern is the RC latency of global connections between large functional blocks which may extend across the whole width of a die [10]. Options to alleviate this problem include optical and RF interconnects, either in free space or through wave guides, both inter and intra-chip [9][10]. According to the 2001 International Technology Roadmap for Silicon

(ITRS), research on RF interconnects should continue until 2007 before appearing in production devices in around 2011 [11].

RF wireless communications will increasingly find applications in all areas of our lives. From devices embedded in the fabric of our houses, to systems which connect us to the wider world both in a social and very real sense; spectrum has now been allocated in the US and Europe for implanted medical instruments, ultra low power radios which monitor the body's own systems and keep remote doctors informed of their status [12]. In the emerging information society, wireless radio communications will be a truly ubiquitous technology.

## 1.2 Wireless Networking Standards

Current wireless data communication standards can be categorised according to the type of network environment they are intended to serve, namely metropolitan, local or personal area networks (MANs, LANs and PANs). There are two MAN standardisation efforts: the IEEE 802.16 WirelessMAN working group and ETSI's Broadband Radio Access Networks (BRAN) project. The IEEE's WirelessMAN standard [13] defines a Medium Access Control (MAC) layer capable of supporting multiple physical, radio, layer types based on single carrier modulation in the 10 - 66 GHz band. Additionally, there is an 802.16a standard supporting single and multi carrier modulation in the 2 - 11 GHz bands. Both IEEE systems are specifically aimed at fixed location, point to multi-point wireless networks. Base stations are deployed with connections to the fixed wired infrastructure. They then service subscriber nodes located on residential and office buildings. This provides a 'last mile' connection similar to xDSL or cable alternatives but at a much higher data rate, potentially up to 134.4 Mbit/sec.

The BRAN project actually consists of four standards: HIPERACCESS, a 25 Mbit/sec fixed link for line of sight (LOS) links above 25 GHz; HIPERMAN for

non-LOS links below 11 GHz; HIPERLAN/2, which provides 25 Mbit/sec links to mobile computing devices; and HIPERLINK, which is intended for very high speed connections at up to 155 Mbit/sec, between multiple HIPERACCESS networks, for example [14]. HIPERMAN has been designed to be compatible with the 802.16a standard and both are supported by the WiMAX Forum, an industry organisation which promotes and certifies broadband wireless access equipment [15]. WiMAX aims to have certified products available in the fourth quarter of 2004. HIPERACCESS is also able to act as the back haul transport for UMTS (3G) cellular radio systems. The benefit of broadband wireless systems is that they are easy to deploy, maintain and are relatively low cost, especially when compared to copper or optical fibre solutions. This makes them ideal where there is no existing infrastructure, in developing countries for instance.

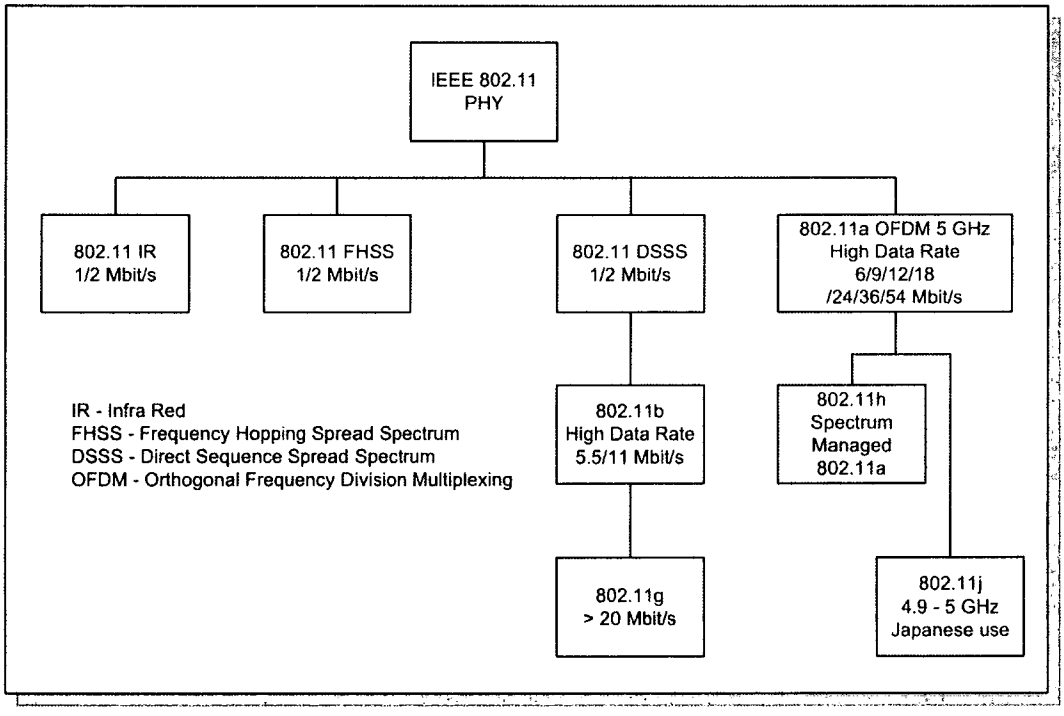


Figure 1.1: IEEE 802.11 Wireless LAN Physical Layer Standards

The principle differences between wireless MANs and LANs are their range and the mobility of nodes. Typically a MAN will work over several kilometers whilst a

LAN is designed to have a range of at most hundreds of metres, usually indoors. HIPERLAN/2 is essentially a LAN standard and complements ETSI's HIPERLAN/1 and the IEEE 802.11 (WiFi) family, of which there are eight members as shown in figure 1.1 [16]. HIPERLAN/1 is designed to provide 20 Mbit/sec connections to portable devices at frequencies in the 5 GHz range [17]. HIPERLAN/2 also uses the 5 GHz band but offers data rates of up to 54 Mbit/sec by employing a multi carrier scheme known as Orthogonal Frequency Division Multiplexing (OFDM) also used by the high rate 802.11a standard [18]. Of the available wireless LANs, 802.11b has the highest market penetration. A rival standard called HomeRF, which was similar to 802.11 FHSS and backed by Intel, Motorola and others, failed at the beginning of 2003 when Intel removed its support in favour of 802.11b [19].

The major advantage of using a wireless LAN is the ability to access other networks at a time and place convenient to the user. The restrictions inherent in desktop PC internet access are removed and new possibilities, such as WiFi 'hot-spots', are created. Hot spots, or WiFi Zones, are areas in which an 802.11 access point is available for internet access. Typical locations are bars, cafes, hotels, airports. A search of the WiFi alliance's database returns 146 separate hot spots in London alone, whilst in New York consumers can access the internet at 51 of the city's McDonald's restaurants (as of November 2003). However, these figures relate only to officially sanctioned sites, usually commercial. Wireless LAN technology has also been taken up by community groups worldwide, but particularly in the United States where it is advocated by `freenetworks.org` and in the United Kingdom by an organisation called Consume. These initiatives help people to set up wireless networks that are local in the traditional sense of 'within their own community'. Whilst such networks, or nodes, are not necessarily connected to the internet, they are a new way of developing a telecommunications infrastructure independently of the traditional telcos.

The 802.11 and HIPERLAN standards exemplify a trend towards greater conve-

nience in technology. This is also displayed by the Personal Area Network standards developed by the IEEE 802.15 working group and the consortium behind the Bluetooth system. PANs consist of short range, low power networks and connections between portable computing devices such as PDAs, PCs, mobile phones, peripherals and consumer electronics [20]. In this context, 'personal' is defined as a space about a person or object that typically extends up to ten metres in all directions and envelops the person while either stationary or in motion [20].

The original technology in this class is the IrDA standard which uses infra-red, line of sight links. It has been available since 1994 and in the current revision of the standard caters for data rates of up to 16 Mbits/sec [21]. However, whilst IrDA is a mature technology, it has a significant weaknesses. It is restricted to line of sight (within 15 degrees) and its point to point protocol makes connectivity inconvenient, restricting its range of applications and usefulness.

The most well known radio based wireless PAN is Bluetooth. The first version of the Bluetooth specification was released in 1999 and updated to the current version 1.1 in February 2001 [22]. It defines the entire system from physical radio layer to outlines of appropriate applications. The lower layers, the radio and the MAC, were standardised by the 802.15.1 task group in June 2002 after the Bluetooth consortium responded to the IEEE's 1998 call for proposals for a wireless PAN standard [20].

Bluetooth overcomes the limitations of IrDA by providing mechanisms for multi-device networks which do not require line of sight to function. There is support for voice and data traffic including an arbitrary data object exchange protocol known as OBEX. This was adopted from an existing IrDA specification. Bluetooth's operating band is the internationally unlicensed Industrial, Scientific and Medical (ISM) allocation at 2.4 GHz, allowing users to travel with their PAN without breaching regulatory requirements. This is also significant from a device manufacturer's viewpoint as products can be sold worldwide without modification, except perhaps for small changes due to language difference.

In addition to the Bluetooth standardisation, the 802.15 group has produced a low range, low power and low data rate wireless PAN standard known as 802.15.4. It was ratified by the IEEE in May 2003 and forms the basis of an industry sponsored specification called 'ZigBee'. This new development is intended to find applications in a very wide range of products, all of which will benefit from low data rate radio communications. Examples are: industrial sensor networks; home automation; health care sensors for monitoring and diagnostics; remote control of consumer electronics; and toys and games with embedded intelligence [23]. ZigBee operates in two frequency bands: at 868/915 MHz and the 2.4 GHz band used by Bluetooth and 802.11b. Like Bluetooth it has a point to multi-point network topology and accommodates up to 255 devices per network, transferring data at a maximum of 250 kbits/sec. Each device is kept as simple as possible. The protocol is designed to support intermittent low data rate traffic from, for example, switches and sensors, giving very long battery lives ranging from several months to essentially infinite (longer than the useful life of the host system) [23].

## 1.3 Conclusion

Many wireless standards will have a role to play in AmI. High bandwidth connectivity is essential but the key components in Ubiquitous Communications are ad-hoc short range networks of which Bluetooth is the only example currently available. An important factor in Bluetooth's success is, and will be, its ability to operate seamlessly in noisy environments. The physical layer radio design is a significant component of that ability. This thesis will examine the specification and design of Bluetooth's radio interface in detail, particularly in light of Bray and Sturman's assertion that strict adherence to the radio specification is not a sufficient criterion for a successful design [24].

### 1.3.1 Bluetooth's Radio Specification

Bluetooth radios transmit and receive data using a type of Frequency Shift Keying (FSK). In FSK systems data symbols are transmitted as slight variations of carrier frequency. Bluetooth uses a positive variation of between 115 kHz and 175 kHz to transmit a binary '1' and negative variation for a binary '0'. The centre of these variations is termed the channel carrier frequency. The Bluetooth Physical Layer Radio Specification defines 80 channels with carrier frequencies of $2402 + n$ MHz where n is an integer channel number between 0 and 79. This channel number changes pseudo-randomly between each data packet to aid immunity to noise and provide a measure of security.

The carrier frequency does not have to be exact, however. Defined in the Bluetooth Radio Specification are transmitter 'modulation characteristics' which allow carrier frequencies different to the ideal integer number of mega-hertz. At the start of each packet transmission an initial error of $\pm$75kHz is permitted. A further drift of 40kHz from whatever the initial value was is then also allowed. These permissible differences from the ideal carrier frequency are an appreciable fraction of the frequency deviations used to modulate, and subsequently demodulate, data. In the case of a 75kHz offset alone the error is up to 65% of the keyed frequency shifts. Whilst these limits on the accuracy of the carrier frequency are specified for a Bluetooth transmitter no limits are placed on the accuracy of a receivers channel frequency generation. This is reasonable as in most, if not all, cases the same circuitry will be used to generate carrier frequencies for transmission and reception. However, as there is no carrier synchronisation in Bluetooth a situation will generally arise where a transmitter has a carrier frequency offset and the receiver has a different offset. Both ends of the radio channel are making estimations of the channel frequency. If the transmitter was to have an offset of greater than 57.5 kHz and the receiver an opposite offset of greater than 57.5 kHz, then the combined error would be at least 115 kHz the minimum shift used to encode data. This would lead

to a failed transmission despite both radios operating well within specification.

Table 1.3.1 summarises some initial frequency errors observed using a real Bluetooth radio. The measurements were made with an Anritsu Bluetooth Test Set monitoring the carrier frequency offsets of a Casira Bluetooth development kit from Cambridge Silicon Radio, the leading manufacturer of Bluetooth transceivers. The results show that initial carrier frequency errors do exist.

| | Mean (kHz) | Maximum Observed (kHz) |
|---|---|---|
| Positive Offsets | 15 | 24.3 |
| Negative Offsets | 25 | 40.5 |

Table 1.1: Measured Initial Carrier Frequency Offsets. CSR Bluecore single chip radio in a Casira development kit, measured with an Anritsu MT8850A Bluetooth Test Set. 10,000 test runs, mean values approximate.

What the results in table 1.3.1 do not show is the effect of carrier frequency offsets on a receiver's ability to successfully demodulate transmitted data. The specification requires that bit error rates do not exceed $10^{-3}$. Schiphorst et al have studied the effect of signal to noise ratio on Bluetooth receivers to determine the minimum SNR needed to achieve the $10^{-3}$ BER [25]. However, there have been no investigations into the combined effect of SNR and carrier frequency errors. In this thesis results will be presented that demonstrate that the performance of a Bluetooth link may degrade significantly when transmitter modulation characteristics are taken into account. Anecdotal evidence from within the Bluetooth design community confirms this observation.

The thesis will go on to investigate how the effect of carrier modulation characteristics can be greatly alleviated through additions to a Bluetooth receiver design.

It is difficult to quantify how the take-up of Bluetooth by the market place was affected by failure to fully take into account all aspects of the Radio Specification. What is certain is that Bluetooth did not succeed as a technology as quickly as predicted. There are many potential reasons for this: an exceptionally complex specification which required significant software development and debug; an excess

of hype leading to unreasonable expectations; competing technologies such as WiFi blurring the rational for Bluetooth and what it was meant to be; and perhaps radio designs which didn't work as well as they might have done. Again, anecdotal evidence strongly suggests that at least one company designing ASICs for Bluetooth did not take into account the effect of carrier frequency errors in their receiver design.

### 1.3.2 Thesis Structure

The emphasis of the thesis is the optimisation of lower power radios for personal area networks by specific reference to the Bluetooth system. Chapter two will consider the shifting definition of personal area networks. In this context, Bluetooth will be examined from a social and commercial viewpoint. Chapter three will move on to give an in depth description of the Bluetooth specification with emphasis on the physical, radio, layer requirements.

In chapter four, an analysis of the transmitted output frequency specification is presented which demonstrated that carrier frequency accuracy and stability have a significant effect upon a Bluetooth receiver's performance. In light of this a method of measuring transmitter frequency characteristics is described in chapter five. The method is specifically aimed at the automatic test equipment used in IC fabrication and was tested on such during an invited presentation at Advantest in Munich.

The effects observed in chapter four are also the stimulus for a Bluetooth receiver architecture presented in chapter six. It is tolerant to carrier frequency inaccuracy and so offers marked performance improvements over more traditional designs.

A prototype Bluetooth radio test instrument, developed in house for use with the School of Engineering's mixed signal IC test station, is described in chapters seven and eight. The design acknowledges the previous work by incorporating the ability to manipulate the carrier frequency of signals presented to a device under test. It also employs a novel software interface that combines test software with Bluetooth protocol software.

Conclusions are drawn in the final chapter.

# Chapter 2

# Wireless Personal Area Networks

## 2.1 Low Power Wireless Networks

The definition of a wireless personal area network has evolved. Previously, a WPAN was taken to refer to connections between devices within a users personal operating space. Typically this meant a ten metre 'bubble' around them. Consequently the types of devices which would be found in a WPAN were those which a user might carry on their person or be used by one person at a time, PDAs and printers for instance. To satisfy the bandwidth requirements of such personal computing resources the data rate of a WPAN was quite low, less than 1 Mbit/sec. This is more than sufficient for voice links, synchronisation tasks and occasional file transfers. Also the power consumption of WPAN radios was consistent with the that of the devices they connected; it is expected that a PDA or cellular phone will require charging once every few days. It is not unreasonable, therefore, to recharge a WPAN radio's battery once every few days. It is into this environment that Bluetooth emerged in 1999. Indeed it could be argued that the definition of a WPAN was largely derived from Bluetooth's original purpose as a mobile, cable replacement technology.

However, current developments are inevitably leading to a shift in the understanding of WPANs. The ZigBee standard, recently ratified as 802.15.4 by the

IEEE, extends its reach beyond that of the traditional WPAN. According to the ZigBee Alliance its initial markets will be in home and building automation, industrial monitoring and control, and wireless sensor networks. The data rates supported by ZigBee reflect these application areas: 250 kbit/sec down to 20 kbit/sec for sensors and automation tasks [23]. Furthermore, low power and low cost solutions are envisioned making embedded ZigBee transceivers feasible for a very wide range of products. The vast majority of these products will not fit into the previous definition of a WPAN. Nevertheless, the standard has been developed by the IEEE WPAN working group.

A third technology in the 802.15 family is a projected standard for a high data rate WPAN. The likely basis of this will be a type of Ultra Wide Band (UWB) radio. However, the parameters of an implementation are still the subject of discussion between two competing industrial consortia, the IEEE and regulatory bodies. UWB has huge potential as it is able to provide large data rates at low radiated power. It therefore enables access to bandwidth intensive services, such as streaming video, to small, portable devices.

A contemporary definition of a WPAN would leave aside emphasis on 'personal' aspects, concentrating instead on the common characteristics of the 802.15 standards: short range, low power, and low cost. It is unavoidable that the phrase WPAN retains ambiguous connotations but, by shifting focus, it is readily seen that WPANs are becoming a first generation of Ubiquitous Communications.

## 2.2 Bluetooth

The emergence of new standards in the WPAN arena has potentially serious consequences for Bluetooth. In previous years Bluetooth has suffered from misconceptions that it is a competitor to WLANs like 802.11b. It can be employed as an access mechanism to other networks, for example forming a link to a laptop PC from a

mobile phone during a dial-up internet session. However, its low data rates mean it is unsuitable as a wired LAN replacement. Consequently, when compared directly to 802.11b it appears to be an inferior technology. This is an unfair judgement; WLANs would be poor choices to add wireless headset capability to mobile phones. Similarly, a current challenge for Bluetooth is maintaining and developing markets in the light of new WPANs.

The new 802.15 standards have distinct capabilities. ZigBee is tailored to simple applications with low data rates and long battery life requirements. UWB, 802.15.3, will provide high data rates. Bluetooth sits in the middle of these extremes. A higher data rate Bluetooth version is being developed [26], perhaps for eventual inclusion in a version 2.0 of the standard. This would compete with lower rate UWB solutions. There are already products which use Bluetooth for automation and control, based upon the CAN and Profibus standards for example. This encroaches upon ZigBee. Modifications to Bluetooth's physical layer to make it more suited to low data rates and lower power consumption have also been mooted. The success of all three technologies is likely to affect the progress of Ubiquitous Computing. It is essential that they each find viable markets whilst maintaining clearly demarcated identities.

When Bluetooth devices first appeared, in 1999 and 2000, sales were lower than expected. This was due to a number of factors including poor interoperation, complicated setup procedures, inflated expectations and the lack of a 'killer application'. Interoperation and setup issues were in part due to the specification's complexity. Over time incompatibilities have been reduced in software protocol stack implementations and in 2002 the Bluetooth SIG initiated a 'Five Minute Ready' campaign to promote ease of use. It is unclear whether a killer application has emerged. However, sales of chips have now surpassed one million units per week and are projected to be ten times that by 2007 [27]. At the same time the average price of a complete chipset will fall from the present level of $6.94 to below $3, achieving the SIG's

original target of $5[1], and generating over $1.5 billion of revenue [27]. A possible driver for this growth is the acceptance of Bluetooth into mobile phones and automobiles. According to market research published in 2003 more than 15 % of new cars built in 2007 will have Bluetooth fitted at the factory [28]. In-car telephony will be their main use, especially in light of new legislation in the UK banning hands-on calls from the beginning of 2004. Other applications could be diagnostic links to engine management systems, safety systems and distribution of audio and video to occupants.

Bluetooth, ZigBee, WiFi and a host of other radiators crowd the 2.4 GHz ISM band. Coexistence between co-located radios is therefore a concern and one which the 802.15.2 task group is charged with addressing. It has developed a method to aid coexistence of Bluetooth and WiFi. This so called Adaptive Frequency Hopping is now part of the latest version of the Bluetooth specification (see section 3.4). In this scheme a Bluetooth radio detects the presence of interference and deliberately avoids that part of the ISM band. The effect is to increase throughput by reducing the number of incorrectly received packets that would otherwise have to be retransmitted on a clear frequency. A more efficient use of the available bandwidth results. Increasing the robustness of Bluetooth receivers in noisy environments would have the same effect, this is the topic of the following chapters.

[1]This often quoted figure was derived from the approximate cost, $10, of a cable Bluetooth replaces, i.e. $5 per radio at either end [24].

# Chapter 3

# Bluetooth Wireless Communications

## 3.1 Ad-Hoc Wireless Connectivity

One of the principle distinguishing characteristics of the Bluetooth system is its ability to form arbitrary, ad-hoc networks with no fixed controlling entity [29]. This flexibility is a result of Bluetooth's usage scenarios, in which many connections, comprising numerous independent ad-hoc networks called *piconets*, may coexist in the same locality. Each piconet can consist of up to eight active Bluetooth devices. Overlapping piconets can be joined together to form a larger *scatternet* so that more than eight devices can be mutually connected, see figure 3.1, with a maximum of ten piconets in a single scatternet [30]. With or without scatternets, a large number of devices can operate within range of each other. The resultant medium access problem is managed with a combined time division duplex (TDD) and frequency hopping (FH) scheme.

In cellular networks, such as GSM, fixed, dedicated base stations control the connection of mobile terminals to the wired communications infrastructure. There is a clear difference between the two wireless link end points. In contrast, Bluetooth

devices are all equal in this regard; there is no equivalent of a base station. Any Bluetooth device can take on the controlling, or *master*, role in a piconet. All other devices in the piconet are *slaves* of the master (denoted by M and S in figure 3.1). Whilst there can be only seven active slaves in a piconet, up to 255 can be virtually connected in a standby or *park* mode and are able to participate again within 2 ms. Two virtually connected slaves are shown in figure 3.1 with dashed outlines.

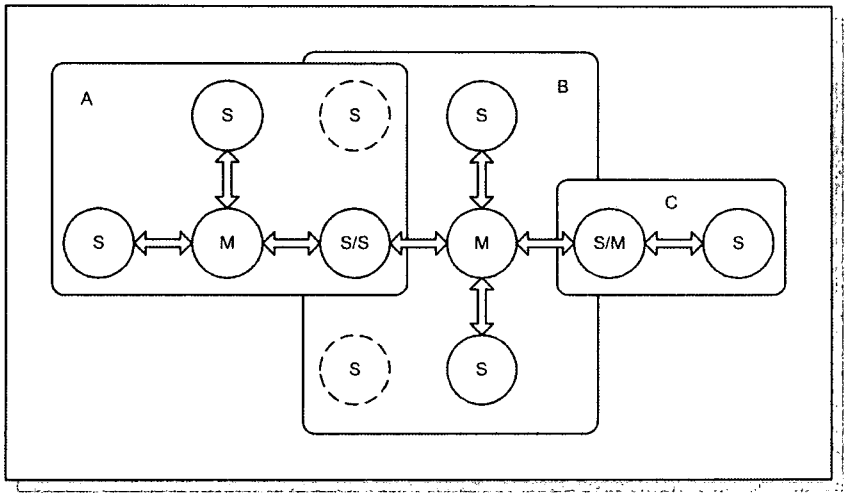


Figure 3.1: Three Piconets (A,B & C) Forming a Single Scatternet

The piconet's master has two important functions: controlling when slaves are allowed to transmit (TDD) and determining the frequency of each transmission (FH). Slaves may only transmit during time windows called *slots*, each of 625 μs and synchronised to the master's clock. Alternate slots are allocated for the master to transmit and receive in. A slave may reply in either the master receive slot immediately after a transmit slot in which it was addressed or during prearranged slots for real time synchronous data. In this manner the master device controls all on air traffic in the piconet. Furthermore, all data must pass through the master even if it is intended for a slave within the same piconet. This gives the star arrangement of connections shown in figure 3.1.

Since piconets may coexist in the same location, there needs to be a mechanism through which the available bandwidth can be shared. Bluetooth uses the interna-

tionally unlicensed ISM band between 2.4 and 2.4835 GHz and splits this spectrum into 79 contiguous channels each of 1 MHz, starting at 2.402 GHz. A piconet's identity can be defined by the particular sequence in which it utilises these channels. In order to comply with regulations, emitters in the ISM band must implement either frequency hopping or direct sequence spectrum spreading techniques. An example of a direct sequence system is IEEE 802.11x (WiFi), whereas Bluetooth is a frequency hopping system.

The transmission frequency hops at a maximum rate of 1600 Hz, or once per 625 μs long slot, in a pseudo random sequence defined by the master's unique Bluetooth address and its clock[1]. In this way the chance of piconets mutually interfering is greatly reduced as the hopping sequence and the starting point in it are uniquely determined by the master. This FHSS provides for a channel which has enhanced immunity to interference, be it from other piconets, other wireless devices or noise sources such as microwave ovens or sodium lamps.

The raw data rate per channel is 1 $Mbs^{-1}$ shared between all devices in a piconet. Theoretically, therefore, the available spectrum can be used at once by 79 separate piconets. In reality this is not possible as there is no coordination of sequences between piconets. Researchers have shown that the incidence of collisions between piconets' frequency choices increases rapidly as the number of coincident piconets increases [31][32]. A practical upper limit is 42 piconets in the same area [33]. However, in the the analysis of [33] it is assumed that any collision of frequency choice between two piconets, at any time during a slot, will cause a failed packet reception. The Radio Specification requires that a receiver is able to resist a co-channel Bluetooth interferer with a signal strength 11dB less than that of the wanted signal (see section 3.3. The analysis in [32] takes this into account and concludes that for co-located piconets (within a 10m diameter room) transmitting the same power,

[1]This maximum rate is sustained only for packets which span a single slot; no frequency change occurs mid-packet for 3 or 5 slot long types. In addition, the maximum hop rate with AFH enabled is 800 Hz as slaves use the same frequency to transmit as that used in the immediately preceding master transmit slot

it is difficult to maintain the 11dB resistance level, and that the main mechanism for avoiding co-channel interference is using a large number of hop frequencies.

## 3.2 Bluetooth Protocol Specification

The Bluetooth specification is split into two parts: a core and a profile specification. The core document describes the protocol layers up to the application layer, see figure 3.2. Profiles are defined to ensure inter-operability by describing how particular types of Bluetooth devices should operate. They include details of what parts of the specification must be implemented, what the user interface should be and what services must be provided to other devices with the same profile. In this way a mobile phone headset for instance will work with any suitably equipped mobile phone regardless of its manufacturer. Devices may implement more than one profile. In the above example it may be desirable for the mobile phone to be able to provide a data link to a laptop computer as well as connect to the headset. These are two quite different tasks and so have different profiles. Conversely, it is not sensible to force the headset, a simple device with low processing power and memory, to implement the parts of the specification which are specific to data connections. Hence profiles enable a consistent, pragmatic application of the core standard.

Inter-operability is crucial to the commercial success of Bluetooth. In addition to profiles, inter-operation is ensured through a rigorously defined protocol specification. To ensure compliance with the specification by all devices, the Bluetooth Special Interest Group (SIG), the body which maintains and develops the standard, requires that new Bluetooth designs pass through a qualification process before they can be marketed as Bluetooth products. Only when a design has been qualified can the IP embedded in the standard be lawfully employed.

Bluetooth has a packet based communications protocol. During any given time slot only one packet can be transmitted in a piconet giving a maximum frame rate of

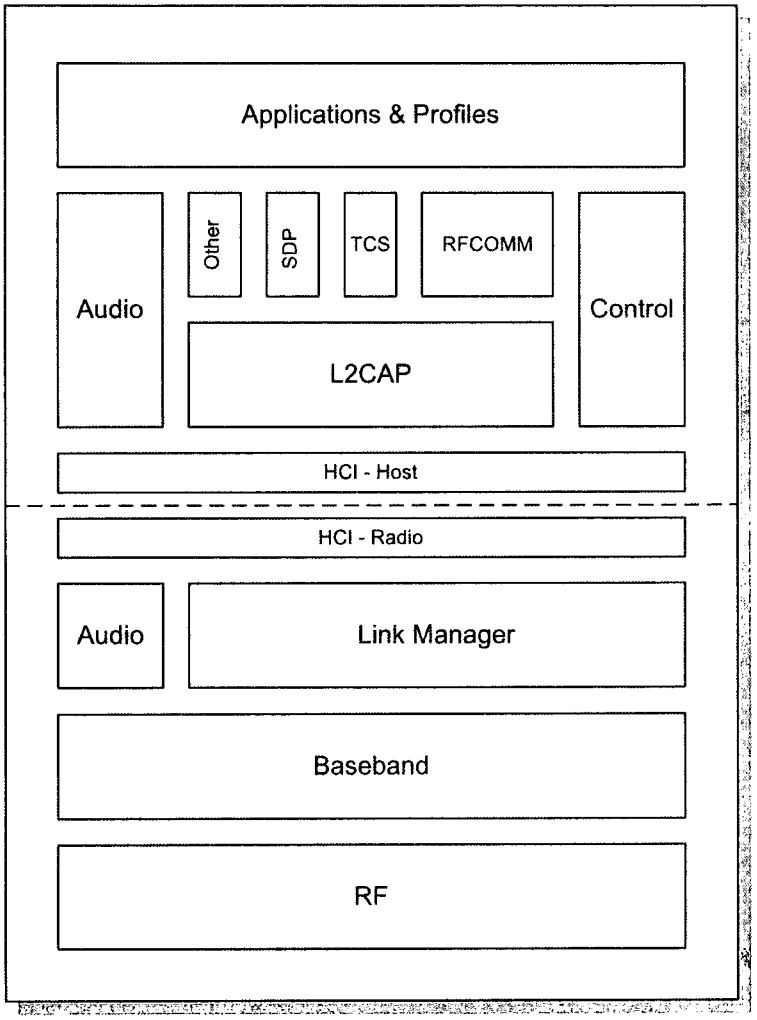


Figure 3.2: Bluetooth Protocol Stack Structure

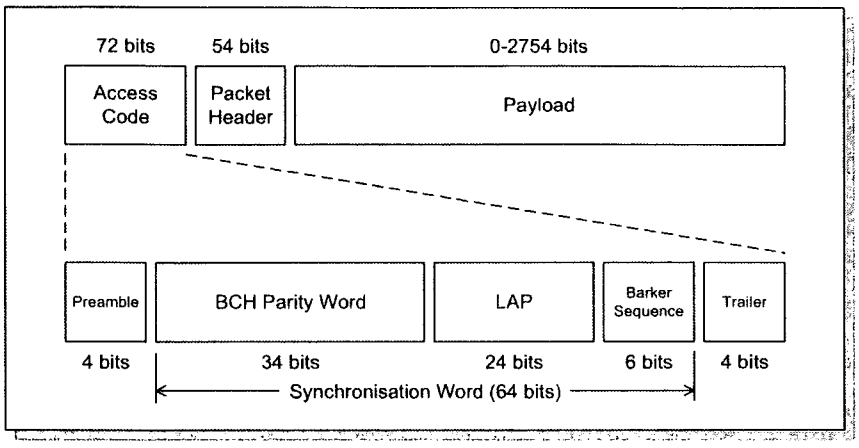


Figure 3.3: Bluetooth Packet Structure

1600 Hz. Packet lengths of 1, 3 and 5 slots are defined, all having the basic structure shown in figure 3.3. Odd numbers of slots are used so that the transmit receive cycle is not disrupted by multi-slot packets. The access code part of the packet identifies the master of a piconet. The synchronisation word is generated from the 24 bit Lower Address Part (LAP) of the master's Bluetooth address. Receivers correlate the packet access code with the correct code for their piconet, ensuring that only those packets are received. If the codes do not match then the radio can be shut down to conserve power. The point at which a slave finds a match is important as it will occur 68 μs after the master began to transmit a packet, hence slaves also use the access code to synchronise their slot timing.

The correlation of access codes occurs in the baseband layer, which is also responsible for maintaining the frequency hopping and selection of the correct packet type. Bluetooth defines several types of packets and the baseband's decision is governed by the type of data to be sent, the amount of data pending in its queue and the current channel conditions. There are three data types: control data, audio or real time sensitive data and non-real time data. All control data packets are one slot long and are used to set up and maintain links. Real time data is conveyed over Synchronous Connection Orientated (SCO) links, in one of four single slot packets called HV1, HV2, HV3 and DV which also contain ordinary data. The difference

between the alternative HV packets is their ability to cope with errors. HV1 packets have a Cyclic Redundancy Check (CRC) field so that errors can be detected and corrupt packets discarded. HV2 and HV3 packets use Forward Error Correction (FEC) at rates of 1/3 and 2/3 respectively, and are therefore more error resiliant. Non-real time data may be transmitted in DH1, DH3, DH5, DM3 or DM5 packets over Asynchronous Connection-Less (ACL) links. In this naming scheme H refers to high rate packets with CRC, M denotes a medium rate packet with 2/3 FEC, whilst the trailing figure is the slot length of the packet. The highest potential throughput is with DH5 packets as they have the smallest relative overhead and no space is taken up with FEC bits. However, they are also the most susceptible to errors and must be retransmitted if incorrectly received. With a BER of 0.4%, two thirds of DH5 packets will be received with errors, reducing the overall throughput [34].

Whilst the low level bit and packet operations are the responsibility of the baseband layer, connection management, setting up links and scheduling traffic falls to the Link Manager (LM). Only SCO link data does not pass through the LM. This is because when a SCO link has been established, by the LM, the transmission times of the data is scheduled into regular slots. Thereafter, all audio data is presented directly to the baseband layer which can support up to three simultaneous SCO links.

Above the LM layer there is a Host Controller Interface (HCI). This layer provides a data transport mechanism between the lower layers in a Bluetooth radio IC and the upper layers which execute on a host processor. The host could be almost any device, a PC, PDA or a mobile phone's native microprocessor for example. Simple applications, such as a headset, only require sufficient intelligence to run the Bluetooth protocol stack with a minimum user interface. In these cases the higher layers will run within the Bluetooth chipset (or chip) and there is no HCI.

Above the HCI there is the Logical Link Control and Adaption Protocol (L2CAP). This layer establishes connections; negotiates quality of service parameters; multi-

plexes data streams from higher layers so they can share a single ACL link and performs segmentation and reassembly of long packets.

The most important of the layers above the L2CAP are RFCOMM, SDP and TCS. RFCOMM is used to provide a RS232 like serial interface so that legacy devices or applications can use Bluetooth transparently. SDP is the service discovery protocol used to unambiguously respond to requests for information about capabilities and profiles supported. TCS, the telephony control protocol specification, defines how telephone calls should be handled by a Bluetooth link. This is important in the context of the Bluetooth's conception as a mobile phone based cable replacement technology.

## 3.3 Radio Specification

The modulation scheme used by Bluetooth is Gaussian Frequency Shift Keying (GFSK), with a symbol duration of 1 μs plus a tolerance of 20 parts per million and a raw bit rate of 1 $Mbs^{-1}$. After subtracting packet overheads, this translates to a useful data rate of 723.2 $kbs^{-1}$ asymmetric and 433.9 $kbs^{-1}$ symmetric. In the symmetric case data is being exchanged at an equal maximum rate between both master and slave. That is both are sending data as fast as possible. In the asymmetric case only one device, either master or slave, is sending large amounts of data with the recipient replying with acknowledgements at a much lower rate of 57.6 $kbs^{-1}$. This capacity is shared between all devices in the piconet [30]. GFSK is described in detail in chapter 5 but in outline it is a particular form of digital FM, in which a 1 is encoded as a positive frequency deviation whilst a 0 is encoded as a negative deviation. The allowed range of deviations is 140 kHz to 175 kHz, with an absolute minimum of 115 kHz. Transmission is in a channel which extends 500 kHz either side of the nominal carrier frequency defined as:

$$f_c = 2402 + n \text{ MHz}, n = 0, \ldots, 79, \tag{3.1}$$

where n is the current channel number in the hopping sequence [22].

Three classes of Bluetooth devices are defined with varying levels of maximum power output. Class 1 devices have a maximum of 20 dBm and have the largest range of 100m in ideal conditions. Class 2 devices may emit up to 4 dBm and Class 3 radios up to 0 dBm , giving a range of less than 10 m. In order to minimise interference from Class 1 equipment, power control for output levels over 0 dBm is mandatory (and optional for other devices). Specific LMP commands are used to keep the power output at the minimum level for successful communication. To support this, receivers are required to implement received signal strength indicator (RSSI) estimation.

### 3.3.1 Transmitter Characteristics

| Packet Type | Frequency Drift |
|---|---|
| 1 Slot Packet | ±25 kHz |
| 3 Slot Packet | ±40 kHz |
| 5 Slot Packet | ±40 kHz |

Table 3.1: Allowable Transmitted Frequency Drift in a Packet

With the frequency of Bluetooth radios hopping at a maximum rate of 1600 Hz, their internal frequency synthesisers must be able to change frequencies and settle within a short time between slots. This settling time is an important parameter. Primarily it determines the amount of time available for the upper protocol layers to be set up for the next slot and for the baseband hardware to calculate the appropriate channel number[2]. A practical upper limit is around 180 µs with values of between 130 µs and 170 µs being common [24]. An immediate consequence of this limit is that it restricts the achievable accuracy of the synthesiser's steady state response. The specification requires that the initial carrier frequency be within ±75 kHz of the value given by (3.1). Therefore, the synthesiser must settle to within this tolerance

[2]This is done with an algorithm, developed by Ericsson, that provides a hopping sequence with the maximum distance (number of channels) between adjacent hops.

in the available time. Additionally, the carrier frequency must not drift by more than the values given in table 3.3.1 and the maximum instantaneous drift rate must be no more than 400 $\mu Hzs^{-1}$ anywhere in a packet. The drift values are not included in the initial $\pm 75$ kHz. These considerations impose stringent requirements on the synthesiser's performance, yet its design must be such that the low cost, low power and small form factor needs of the overall system are not compromised. The type of synthesiser chosen for those radio designs where information was available, is a PLL with programmable VCO. The synthesiser (VCO) programming word is derived from a register bank, defined in the Baseband Specification, and addressed by the output of the Baseband Hop Selection Kernel.

### 3.3.2 Receiver Characteristics

| Requirement | Ratio |
|---|---|
| Co-Channel interference | 11 dB |
| Adjacent (1 MHz) interference | 0 dB |
| Adjacent (2 MHz) interference | -30 dB |
| Adjacent ($\geq$ 3 MHz) interference | -40 dB |

Table 3.2: Receiver Interference Performance, Signal to Interference Ratios

The minimum raw bit error rate (BER) is defined as $10^{-3}$ (0.1%). Receivers must achieve this BER at an input power level of -70 dBm . They should also be resistant to the levels of interference, given in table 3.3.2. The values correspond to a wanted signal at -60 dBm and the interfering signals are Bluetooth modulated. In essence, this part of the specification is to ensure that piconets can coexist effectively and is defined in conjunction with limits on the transmitted in-band spurious emissions. Receivers must also be tolerant to signals of given maximum strengths outwith the ISM band.

## 3.4 Adaptive Frequency Hopping

Adaptive Frequency Hopping (AFH) is used to improve the performance of physical links in the presence of interference and to reduce the interference experienced by other devices, of whatever type, in the ISM band. AFH classifies radio channels as either good, bad or unknown. The precise method used to assess a channel is undefined in the specification. However, assessments are made locally and may be performed either by a host and based upon measurements of throughput, or by the radio device using an interference level metric. Information about each channel is summarised in the 'AFH Channel Map' which is then used by the Baseband Adaptive Hop Selection Kernel to make transmission frequency decisions. Control over the use of AFH is the responsibility of a piconet master device which sends individual slaves special Link Manager PDUs containing AFH information. These PDUs contain the AFH Channel Map for the piconet and a time at which a new AFH mode is to be enabled. AFH is only valid in 'Connected Mode'. Other modes such as 'Inquiry Scan' and 'Page Scan' which are used to set up connections are not affected.

AFH does increase the complexity of a Bluetooth device. Additional logic is required to implement the Adaptive Hop Selection Kernel in the Baseband Layer, AFH Link Manager Protocol functions and channel assessment algorithms, although the latter may be performed by the Host. The Physical, radio layer is not affected, however, as AFH changes only the range and order of channel selections and not their individual frequencies.

## 3.5 Bluetooth Radio Test Specification

Test procedures for the Bluetooth RF layer are defined in two documents: the Bluetooth Test Mode section of the core specification [22] and the RF Test specification [35]. Figure 3.4 shows a diagram of the required RF test setup. In an RF test two

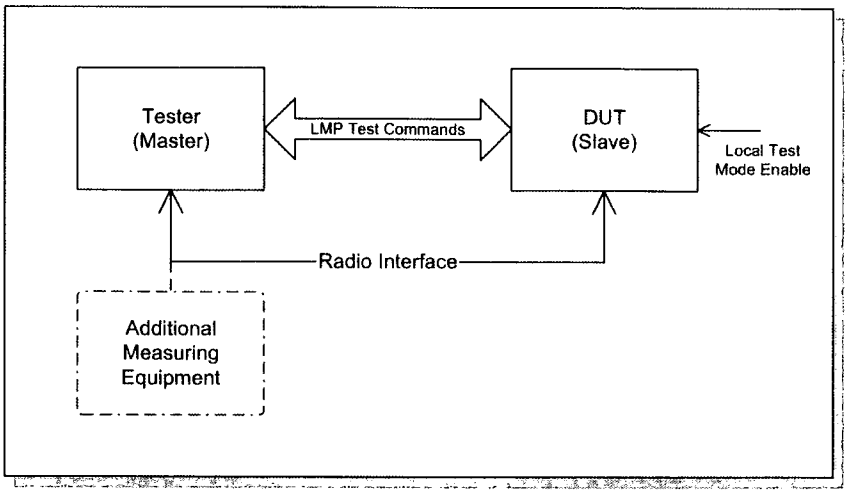


Figure 3.4: RF Test Setup

Bluetooth devices are involved: the device under test (DUT) and the tester. Together these form a two radio piconet with the tester as master. Control of the slave DUT throughout the test is via LM protocol commands sent over the air interface.

To initiate a series of tests the master transmits an LMP test control command packet. If the DUT is able to, it will respond with an LMP Accepted message. A DUT must have its test mode locally enabled, by a hardware or software signal, in order to enter a test mode. This is because some of the RF tests require the DUT to become non-compliant with the regulations governing ISM band usage. It is important that devices cannot enter into test mode inadvertently.

The test specification exactly defines the packet payloads. Patterns used are all 0's, all 1's, ...101010..., ...11110000... and a 9 bit pseudo random sequence (PRBS9) which starts with nine 1's. Whilst in test mode, data whitening is therefore deactivated. Data whitening is a channel coding scheme which ensures that long sequences of the same bit are not transmitted. It is used to ensure that the transmitted carrier frequency does not experience large drifts. Test packets have the format shown in figure 3.3 but with a predefined payload.

Two general types of test are defined in test mode: transmitter tests and loopback tests. In loopback tests the master sends packets to the DUT which decodes

them and retransmits a copy. This allows the master to exercise the DUT's CRC, FEC and header error check (HEC) mechanisms by deliberately introducing errors into the packets and analysing the responses. In transmitter tests, packet payload type, length, frequency and transmission interval are determined by the master and communicated to the DUT through LMP commands. The master then sends POLLs[3] to the slave which responds with the predefined test packet. In these tests the DUT's conformance to the radio specification is verified.

## 3.6 Conclusion

Strict adherence to the specification is not necessarily a sufficient criterion for a successful Bluetooth design. Bray and Sturman [24] state that:

> "...several of the [radio performance] parameters are only acceptable on paper and in particular do not address the real-world scenarios in which Bluetooth is to be employed."

Potentially, one such parameter is the tolerance on the transmitted radio frequency carrier. The specification requires that the initial carrier frequency be within 75 kHz of the correct value. However, this is an appreciable fraction of the allowed range of frequency deviations used to encode data. A hypothesis is therefore that carrier frequency errors which are within tolerance may, nevertheless, have a significant detrimental impact upon received bit error rates. This will be explored further in chapter 5 after a review of Gaussian Frequency Shift Keying in the following chapter.

[3]Short packets which require the addressed slave to respond in the next master receive slot.

# Chapter 4

# GFSK: Principles and Demodulation Methods

## 4.1 Gaussian Frequency Shift Keying

GFSK is a member of a generic class of constant amplitude, angle modulation schemes, which may be defined by [29]:

$$s(t) = \sqrt{\frac{2E}{T}} \cos(2\pi f_c t + \phi(t, \alpha)) \,, \tag{4.1}$$

where $E$ is the bit energy, $T$ the symbol duration and $f_c$ is the carrier frequency. Transmitted information is contained in the phase (angle) term:

$$\phi(t, \alpha) = 2\pi h \sum_{i=-\infty}^{\infty} \alpha_i q(t - iT) \tag{4.2}$$

where $\alpha_i$ is the message signal and

$$q(t) = \int_{-\infty}^{t} g(\tau) \mathrm{d}\tau \,. \tag{4.3}$$

The function $q(t)$ describes the shape of the phase transitions whose magnitude is proportional to the modulation index $h$.

In simple binary FSK (BFSK) $\alpha_i$ may take two values, either -1 (usually representing a binary zero) or 1, and $q(t) = \frac{1}{2}$. This gives rise to linear phase transitions (from (4.2)) of $\pm h\pi$ [29][36]. Hence if the phase at the beginning of a symbol is $\phi(0)$, a binary zero is encoded as a frequency $f_0$ and a binary one is encoded as a frequency $f_1$.The phases after transmissions of a binary zero and a one are then respectively:

$$\begin{aligned} \phi(\alpha_0) &= \phi(0) - h\pi \\ &= \phi(0) + 2\pi f_0 T \end{aligned} \tag{4.4}$$

$$\begin{aligned} \phi(\alpha_1) &= \phi(0) + h\pi \\ &= \phi(0) + 2\pi f_1 T\ . \end{aligned} \tag{4.5}$$

Assuming that $f_1 > f_0$ the phase transition from a zero to a one is:

$$\begin{aligned} \phi(\alpha_1) - \phi(\alpha_0) &= h\pi - (-h\pi) = 2\pi h \\ &= 2\pi f_1 T - 2\pi f_0 T = 2\pi(f_1 - f_0)T \end{aligned}$$

therefore:

$$\begin{aligned} h &= (f_1 - f_0)T \\ &= 2f_d T \text{ where} \end{aligned} \tag{4.6}$$

$$f_d = (f_1 - f_c) = (f_c - f_0) \tag{4.7}$$

That is the modulation index $h$ is defined in terms of the peak frequency deviation $f_d$ from the carrier frequency $f_c$ and the symbol duration $T$ [36]. It is similar to the modulation index $\beta$ used in analogue FM, where $\beta$ is defined as the ratio of maximum carrier frequency deviation to the maximum frequency of the modulating signal. However, in FSK, h is the ratio of maximum carrier frequency deviation to half the symbol rate of the modulating data stream.

The definition of h given by 4.6 is used for all forms of FSK, regardless of $q(t)$,

including GFSK and the Fast FSK or Minimum Shift Keying (MSK), first described in [36] which has $h = \frac{1}{2}$ and is used by the GSM mobile telephony standard.

### 4.1.1 Pulse Shaping

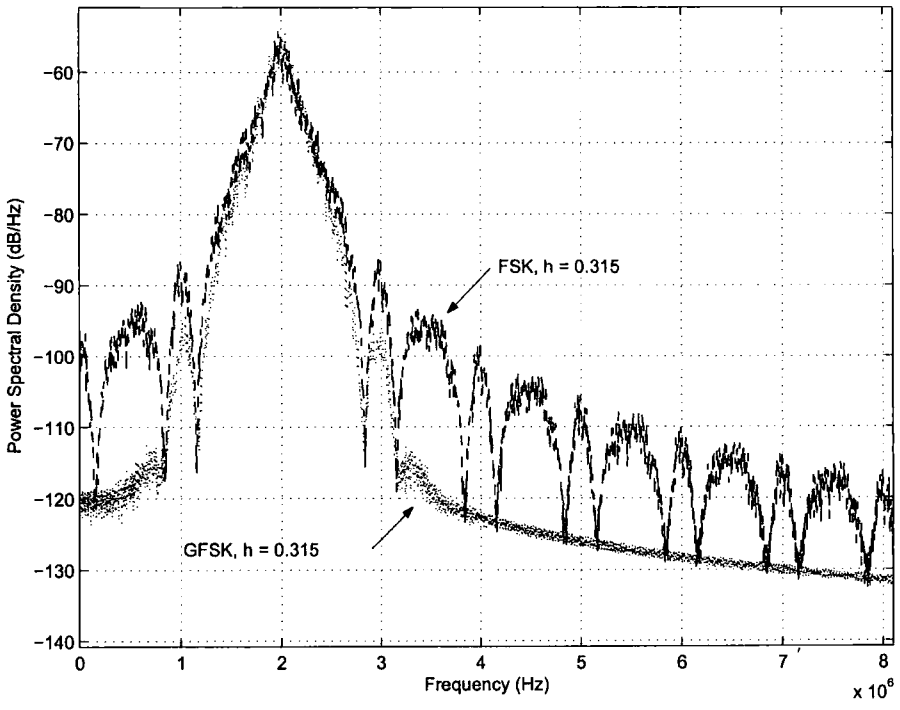


Figure 4.1: PSD of FSK and GFSK Modulated Data.

The spectral occupancy of standard FSK is large. Figure 4.1 shows that for a BFSK signal with a symbol rate of 1 Mbaud, the first modulation side lobe is 30 dB down from the main signal and offset by 1 MHz. For maximum use of bandwidth in a multichannel communications scheme it is desirable to space channels as closely as possible. If in this case the channel spacing was coincident with the symbol rate, then co-channel interference would occur as the adjacent channels would coincide with the side lobes at 1 MHz and 3 MHz . Some method of restraining the bandwidth of the modulated signal is therefore required.

Conversely, when rectangular pulses are passed through band limited channels, inter symbol interference (ISI) occurs as the pulses tend to spread in time. Increasing

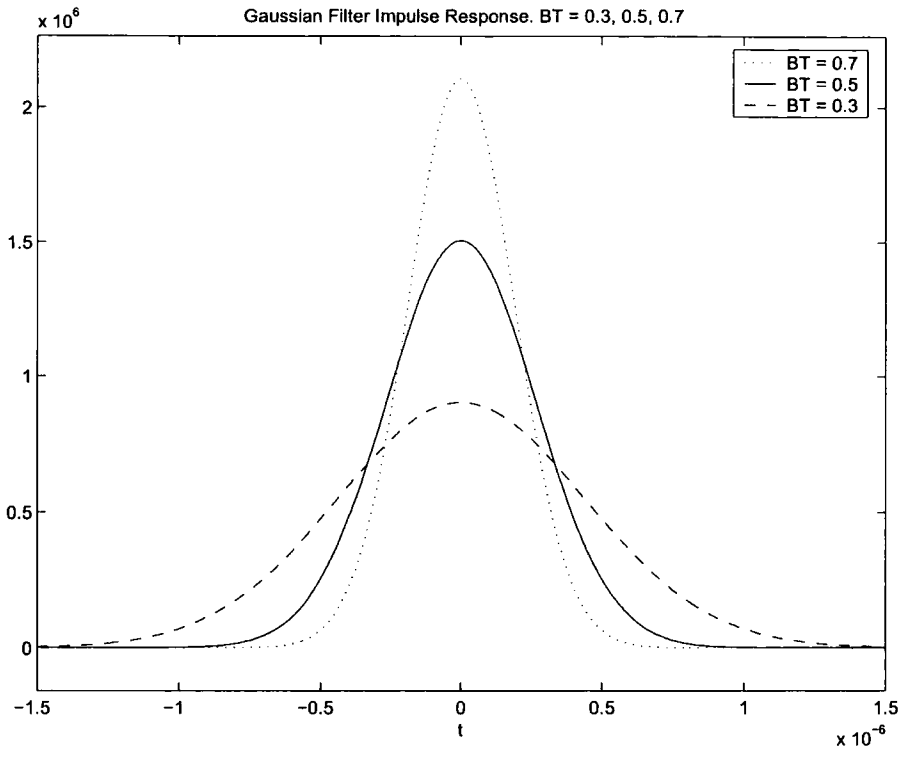


Figure 4.2: Gaussian Filter Impulse Responses.

channel bandwidth to minimise ISI is therefore also desirable [2]. A modulation technique which reconciles the conflicting requirements of spectral occupancy and ISI reduction is therefore desirable.

Channel bandwidth could be constrained by filtering at RF. However this is not practical as the channel frequency tends to be variable and the filter would need an extremely narrow response. An alternative is to alter the shape of the baseband phase transitions, that is modify $q(t)$, by filtering the message signal before modulation. This is known as premodulation filtering or pulse shaping.

A bandwidth efficient pulse shaping scheme that completely eliminates ISI was first proposed by Nyquist [37]. Its filter has an impulse response in the form of a $\frac{\sin(x)}{x}$ function centred on the current symbol with its nulls at integer multiples of the symbol duration. Hence samples at the centre of a symbol have no components from any other symbol. However, to achieve such an impulse response an ideal brick-wall filter is required in the frequency domain. This is not practically achievable so

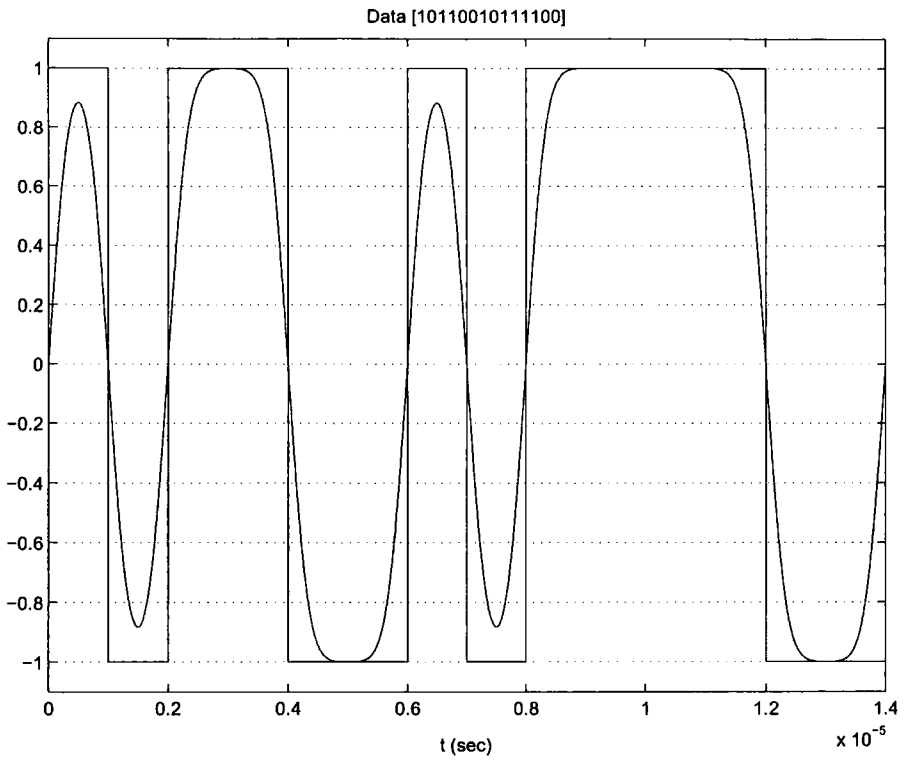


Figure 4.3: Effect of Gaussian Premodulation Filter.

approximations such as the Raised Cosine Roll-off filter are used [2][38].

Unfortunately, Nyquist pulse shaping is not appropriate for systems such as Bluetooth. To benefit from the ISI cancellation exact synchronisation between transmitter and receiver is required; this is not the case in Bluetooth. Also, it is essential that the pulse shape is preserved throughout transmission which cannot be guaranteed with non linear, power efficient class C amplifiers. Such amplifiers are an important component of low power low cost systems. Indeed, FSK is well suited to them due to its constant amplitude characteristics. An alternative pulse shaping filter, which has good bandwidth limiting properties whilst introducing some ISI, is therefore used. This filter has a Gaussian impulse response, hence GFSK, and was first suggested by Murota and Hirade [39] for MSK.

Gaussian filters have an impulse response $h(t)$ given by:

$$h(t) = B\sqrt{\frac{2\pi}{\ln 2}} \exp \frac{-2\pi^2 B^2 t}{\ln 2}, \tag{4.8}$$

where $B$ is the filter's 3 dB bandwidth, and are characterised by their $BT$ product, $T$ being the baseband symbol duration. Figure 4.2 graphs $h(t)$ for three values of $BT$, with $T = 1\mu s$, Bluetooth uses $BT = 0.5$. The figure also illustrates ISI. If the centre of the current symbol is at $t = 0$ then it extends to $t = \pm 0.5\mu s$. However, the filter impulse response extends to approximately $t = \pm 1\mu s$ for $BT = 0.5$. Consequently, a baseband rectangular pulse will be smeared into the previous and next symbols. This is also shown by figure 4.3; the amplitude and shape of a symbol in the filtered data is influenced by the adjacent symbols.

Referring back to figure 4.1, the effect of the Gaussian pulse shaping on the modulated signal's bandwidth can be seen[1]. There is a 10 dB reduction in the amplitude of the first side lobe and a 30 dB reduction in the second side lobes. Co-channel interference is much reduced at the expense of some ISI.

## 4.2 GFSK Demodulation

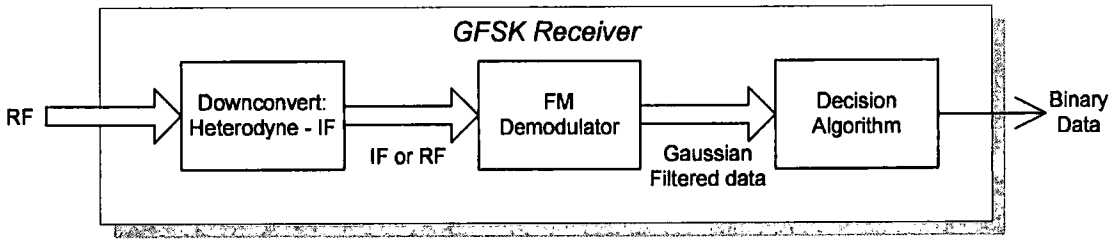


Figure 4.4: GFSK Receiver Structure.

Demodulation of a GFSK signal and recovery of the transmitted digital message is achieved in either two or three stages (figure 4.4). In either case a FM demodulation technique is used to extract the gaussian filtered NRZ signal, before some form

[1]Both power spectral density plots used the same data and modulation index. The premodulation filter had $BT = 0.5$.

of decision algorithm is employed to determine whether a 1 or 0 was sent during each bit interval. The input frequency to the FM demodulator is either the full RF or an intermediate frequency (IF). In the RF case the receiver is said to be homodyne or direct conversion and the demodulator includes mixing elements to directly convert the input signal to zero IF, or baseband, frequencies. A heterodyne receiver, on the other hand, has an initial down conversion to a non zero IF, typically of a few MHz and the demodulator always acts upon signals at this frequency.

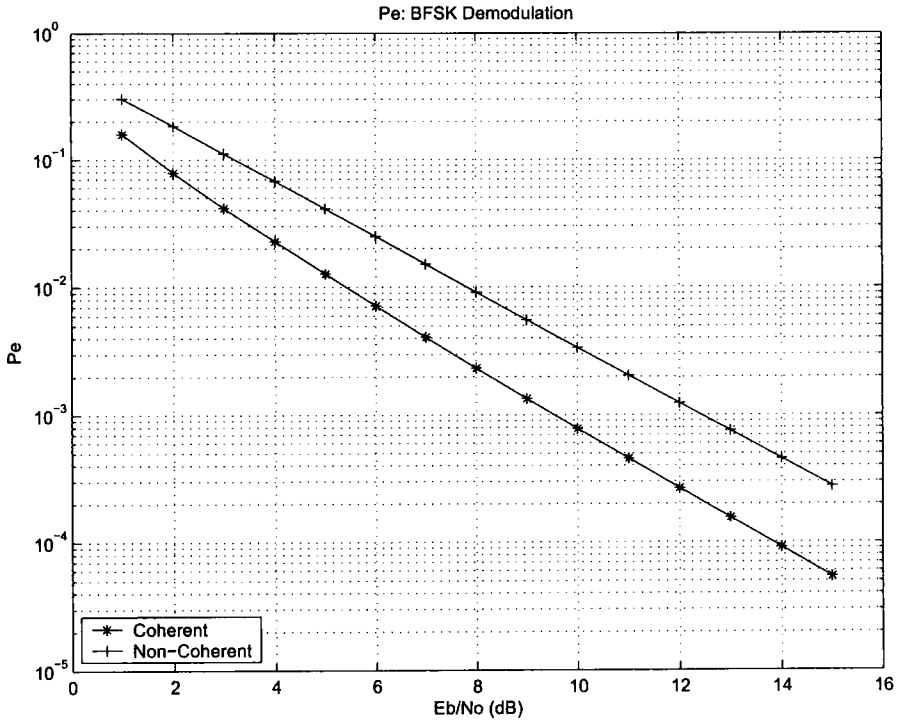


Figure 4.5: Demodulation of BFSK - Theoretical Minimum Probability of Error.

The demodulation may be either coherent or non-coherent. In coherent detection the precise transmitted carrier frequency and phase is known at the receiver [2] and demodulation proceeds via matched filter detection [2][38][40]. Information about the carrier can either be transmitted as a pilot tone or be recovered at the receiver using, a Costas loop for example [41]. Other carrier synchronisation schemes have been proposed, initially by deBuda [36] for MSK where the peak frequency deviation is a quarter of the symbol rate, that is $h = \frac{1}{2}$. Carrier revovery schemes are discussed

further in chapter 6.

For binary FSK the probability of error for coherent reception is given by (4.9)[2], which is graphed in figure 4.5 along with the corresponding curve for non coherent reception (4.10)[2].

$$P_{e,C} = Q\left(\sqrt{\frac{E_b}{N_o}}\right) \tag{4.9}$$

$$P_{e,NC} = \frac{1}{2}\exp\left(-\frac{E_b}{2N_o}\right) \tag{4.10}$$

As can be seen, non-coherent detection incurs a performance penalty of approximately 1.5 dB but it reduces the receiver complexity and is most appropriate for frequency hopping communications systems with bursty traffic such as Bluetooth [42].

### 4.2.1 GFSK Demodulation Algorithms

Two non-coherent demodulation algorithms are discussed, the quadrature detector and phase shift discriminator. Both are well known forms of FM demodulator, the phase shift discriminator being a more complex implementation of the classical, analogue, limiter-discriminator circuit suited to a homodyne architecture.

#### 4.2.1.1 Quadrature Detector

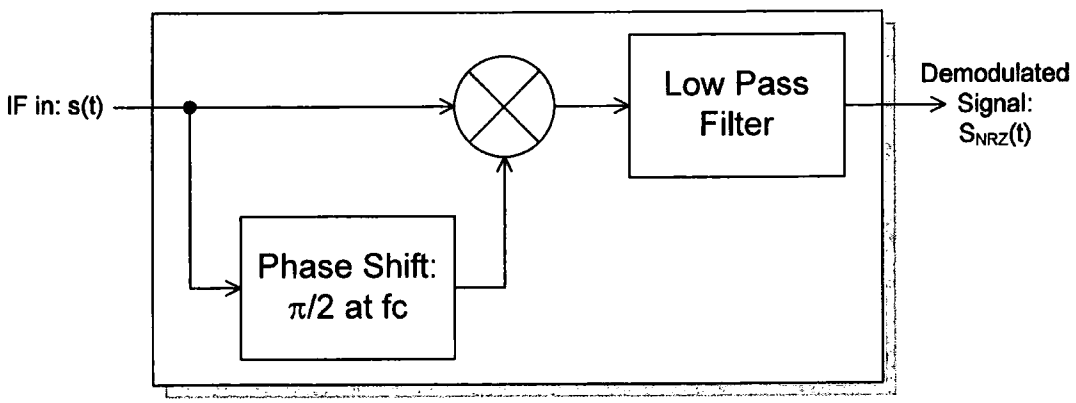


Figure 4.6: Quadrature Detector.

Quadrature detection, also known as FM discrimination and FM-AM conversion, is one of the most popular forms of FM demodulation due to its ease of implementation in an IC at low cost [2][25]. The goal here, as with any FM demodulator, is to provide an output which is proportional to the instantaneous frequency of the input signal. This is achieved as shown in figure 4.6. The Quadrature Detector is so called as it uses a $\frac{\pi}{2}$ phase shift network to create a quadrature copy of the input modulated signal in the first stage of demodulation. If the frequency response of the phase shift block is:

$$\begin{aligned} \phi(f) &= -\frac{\pi}{2} + 2\pi K(f - fc) \\ \phi(f_c) &= -\frac{\pi}{2} \end{aligned} \tag{4.11}$$

then its output with a GFSK input will be:

$$s(t) = A(t)\cos(2\pi f_c t + h\int_{-\infty}^{t} m(\tau)\mathrm{d}\tau) + \phi(f_i(t)) \tag{4.12}$$

where $f_i(t)$ is the instantaneous frequency of the GFSK signal. Multiplying $s(t)$ with the original IF signal and low pass filtering yields an output proportional to the transmitted NRZ message signal $m(t)$:

$$\begin{aligned} s_{NRZ}(t) &= \frac{A}{2}\cos(\phi(f_i(t)) \\ &= \frac{A}{2}\cos(-\frac{\pi}{2} + 2\pi K(f - fc)) \\ &= \frac{A}{2}\sin(2\pi K h m(t)) \\ &= Cm(t) \text{ for small phase changes.} \end{aligned} \tag{4.13}$$

In a digital implementation the phase shift can be produced by delaying the input signal by an integer number of samples. The number of samples required, to give a $\frac{-\pi}{2}$ phase shift at $f_c$, is an odd multiple of $T_s = \frac{f_s}{4f_c}$, where $f_s$ is the sampling frequency of the IF input data [25]. An analogue version would employ a RLC

tank circuit [43]. Quadrature detection is well suited to a heterodyne architecture, performing the function of the central block in figure 4.4, where the initial down-conversion stage uses a locally synthesised LO appropriate to the current channel number in a frequency hopping system [44].

#### 4.2.1.2 Phase Shift Discriminator

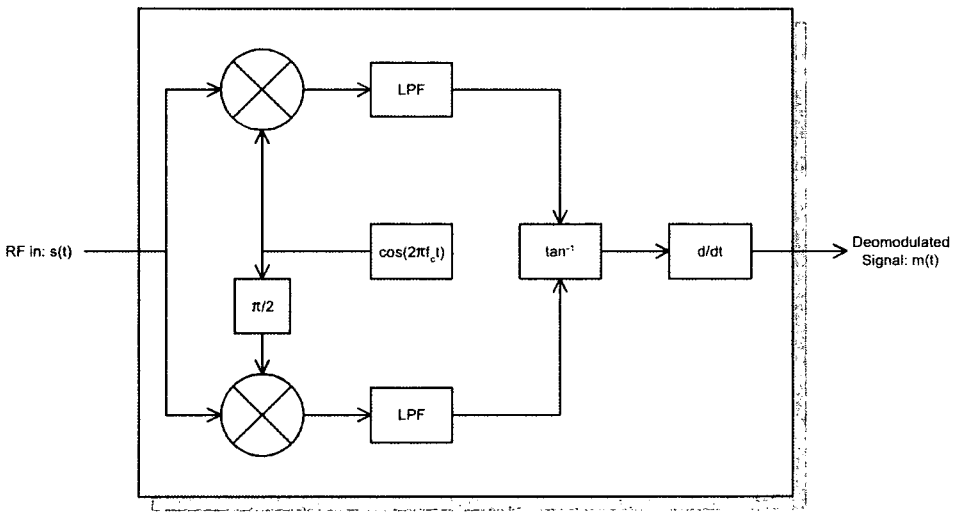


Figure 4.7: Phase Shift Discriminator.

In contrast to the quadrature detector, phase shift discriminators (figure 4.7) are usually seen in direct conversion homodyne architectures. The phase shift discriminator derives it name from its basic operating principle; demodulation is achieved by tracking the phase shift of the input signal relative to that of a locally generated copy of the nominal carrier frequency. The received RF signal is first converted to a complex baseband signal (4.14) and (4.15), by mixing with a LO at the nominal carrier frequency $f_o$.

$$s_I(t) = \frac{A(t)}{2}[\cos(4\pi f_c t + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau) - \cos(2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau)] \tag{4.14}$$

$$s_Q(t) = \frac{A(t)}{2}[\sin(4\pi f_c t + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau) - \sin(2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau)] \tag{4.15}$$

The resulting In-phase and Quadrature signals are then low pass filtered to remove the frequency components at $2f_c$, before the absolute phase is extracted by arctan and phase unwrapping operations to leave:

$$\phi(t) = h \int_{-\infty}^{t} m(\tau) \mathrm{d}\tau \tag{4.16}$$

The Gaussian filtered message signal is then retrieved by differentiation:

$$m(t) = \frac{\mathrm{d}(\phi(t))}{\mathrm{d}t} \tag{4.17}$$

Differentiation can be performed digitally either by subtracting each sample from the next or by passing $\phi(t)$ through a linear phase filter with a transfer function $H(\omega) = j\omega$.

A disadvantage of the quadrature detector is its dependance upon both the phase and amplitude of the incoming signal. The phase shift discriminator uses only the signal phase and consequently is slightly better: according to [25] approximately 1.5 dB less SNR is required to achieve a particular BER.

### 4.2.2 GFSK Decision Algorithms

Decision algorithms may take a variety of forms but the simplest type is known as *integrate-and-dump* (IaD). With IaD the samples of each demodulated symbol are summed and the total is compared to a threshold value, zero for NRZ signalling. In the following discussions a 1 bit IaD, or *bit slicer*, will be used with a threshold value of zero. This algorithm compares the central sample of a symbol to zero and decides that a 1 was transmitted if the sample is greater than zero and that a 0 was transmitted otherwise.

Other decision algorithms attempt to mitigate for the effects of ISI using adaptive equalisation. The three types of equalisation are the Linear Transversal Equaliser (LTE), the Decision Feedback Equaliser (DFE) and the Maximum Likelihood Se-

quence Estimator (MLSE) [45]. The most popular of these is the DFE for its compromise between complexity and effectiveness, although there are designs which employ a MLSE, [46] for example which reports a 5.3 dB advantage over a simple bit slicer. DFE operates in the context of GFSK by modifying the sample values of the current symbol to correct for the ISI effect of the previous symbol and, in some designs, an estimate of the next symbol. A performance increase of 1.5 dB was observed by Schiphorst et al [25] for DFE over IaD when considering 8 samples per symbol. When 1 sample per symbol was used there was no benefit from DFE due to the reduced amount of information per symbol.

# Chapter 5

# GFSK Demodulation Algorithm Performance

This chapter discusses the effects upon bit error rates of frequency errors, in the form of offsets and drifts, introduced into a GFSK modulated carrier frequency. The discussion is in the context of a Bluetooth radio and so the errors considered are those allowed by the Bluetooth specification (see section 3.3).

In order to investigate the effects of carrier frequency errors a model of a GFSK transceiver was developed in Matlab (see Appendix A for relevant extracts of the source code). The following sections describe the model and present the results of simulations of the quadrature detector and phase shift discriminator's carrier frequency error performance.

The simulations presented here model the effect of carrier frequency errors upon the ability of GFSK receivers to successfully demodulate data in the presence of noise. The effect of Doppler frequency shifts, caused by the relative velocity of transmitter and receiver, is not explicitly addressed. Given the use of Bluetooth devices in PANs Doppler shifts will be at most tens of hertz and will anyway appear as additional carrier errors at the receiver. The effects of multi-path signals are not taken into account. Filter quantisation effects are also not taken into account in the

presented results.

## 5.1 Matlab Simulation

### 5.1.1 Transmitter Model

Carrier frequency errors are modelled by introducing two additional terms (underlined) into the general equation of a GFSK signal ( (4.1) – (4.3)) to yield:

$$s(t) = \cos(2\pi f_c t + \underline{2\pi f_o t + 2\pi\alpha(t)t^2} + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau). \tag{5.1}$$

A carrier frequency offset $f_o$, is simply a deviation from the nominal channel frequency $f_c$. By definition this offset is the error at the beginning of a transmission and so is constant throughout. In contrast carrier drift $\alpha(t)$ is by definition zero at the start of a packet and may be an arbitrary function of time. If $\alpha(t) = K$ then K is a constant drift rate which will result in a quadratic phase trajectory. A linear frequency chirp will be superimposed upon $f_c$, $f_o$ and the frequency deviations due to modulation.

The complete transmitter model has four parts. Firstly, a data vector of a specified length is generated with a uniform random process. These bits are then up sampled to a rectangular waveform with a basic period of 1 µs , the Bluetooth symbol duration, at the simulation sampling frequency $f_s$. There are then $\frac{f_s}{1\mu s}$ samples per symbol. The resulting vector is then convolved with an FIR filter with a Gaussian impulse response and finally the filtered data is modulated on to a carrier frequency of 2 MHz.

The pre-modulation FIR filter's coefficients are generated by sampling h(t) from (4.8) with $BT = 0.5$ at linearly spaced intervals. The filter length was found empirically to be 640 for a 3 dB bandwidth of 500 kHz at a simulation sampling frequency of 320 MHz. The filter impulse response is truncated to two symbol lengths [47](see

figure 4.2).

The final modulation is performed by a modified version of Matlab's VCO function according to the following equations:

$$s_{Tx}(t) = \cos(2\pi f_c t + 2\pi\alpha t^2 + k_f \sum_{i=1}^{N} m(i)) \quad (5.2)$$

$$k_f = \frac{h}{2T}\frac{2\pi}{f_s} \quad (5.3)$$

$$f_c = (2 + \text{channel number}) \times 10^6 + f_o \quad (5.4)$$

$$N = (\text{number of bits})\frac{f_s}{T}, \quad (5.5)$$

and $m(i)$ is the vector of sampled, Gaussian filtered data. Input variables to the transmitter model are therefore the data length in bits, modulation index $h$, channel number (0-79), carrier frequency offset $f_o$ Hz and carrier frequency drift rate $\alpha$ Hzs$^{-1}$.

### 5.1.2 Channel Model

The ISM radio channel is simply implemented as an additive white gaussian noise (AWGN) channel with zero fading. The noise magnitude is calculated according to [48] to achieve the correct $\frac{E_b}{N_o}$; a receiver noise bandwidth of 1 MHz is assumed and modelled by passing the noisy GFSK signal through a 512 tap FIR bandpass filter centered at $f_c$ [25].

The up and down conversion processes in the simulated transmitter and receiver are assumed to be ideal and were eliminated in the simulation. The results are therefore of back-to-back IF performance.

### 5.1.3 Receiver Models

Both of the demodulator types described above are implemented. The quadrature detection algorithm uses an intermediate frequency, $f_c$ of 2 MHz, a delay of 40 samples to implement the phase shift and a Chebyshev Type II low pass filter with

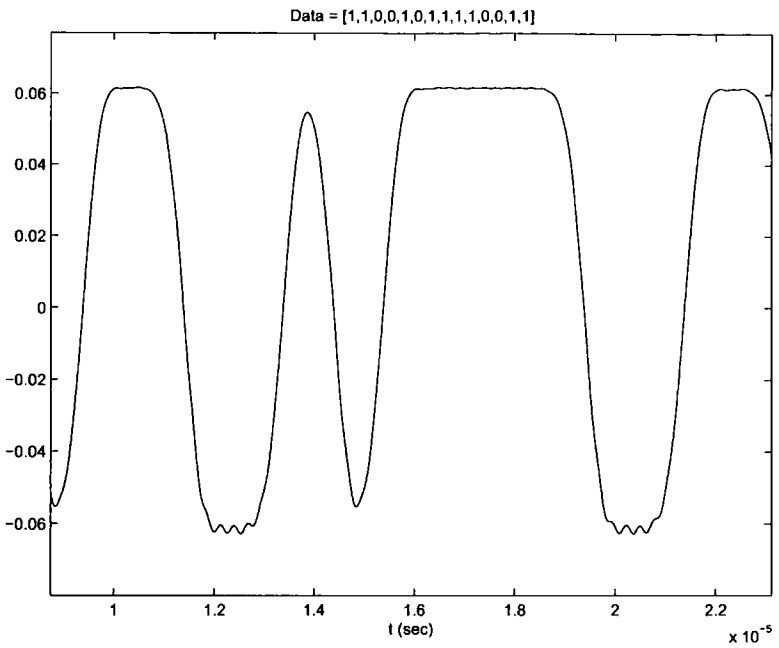


Figure 5.1: Quadrature Detector Output

a 1 MHz pass band. The phase shift discriminator also has an input frequency of 2 MHz (channel 0) and the differentiation is performed by a 1001 tap Parks-McClellan FIR filter. Figure 5.1 illustrates the quadrature detector output for data transmitted on channel 0 with $f_o = 0$ and $\alpha = 0$. Data is recovered from the gaussian filtered NRZ signal by a slicing detector.

### 5.1.4 Simulation

Both demodulation algorithms were tested with initial carrier offsets of 0 to 70 kHz in 10 kHz steps and carrier drift rates of 0 to 14 Hz$\mu$s$^{-1}$ in 3.5 Hz$\mu$s$^{-1}$ steps. The length of data in each run of the simulation was 2870 symbols which is equivalent to a DH5 packet. Thus the maximum drift rate used corresponds to the allowed carrier drift across a 5 slot packet, i.e. $14\text{Hz}\mu\text{s}^{-1} = \frac{40\text{kHz}}{2870\mu\text{s}}$.

Simulations were run with signal to noise ratios ($\frac{E_b}{N_o}$) of 11 dB to 30 dB in 0.5 dB steps for the quadrature detector and 11 dB to 35 dB in 1 dB steps for the phase shift discriminator. At each SNR value 100 iterations of a *modulate packet - add*

*noise - demodulate packet - compare* cycle were carried out. This procedure resulted in a total of 52000 simulation cycles for the quadrature detector and 45400 cycles for the phase shift discriminator.

## 5.2 Results

### 5.2.1 Quadrature Detector

Figures 5.2,5.3 and 5.4 show the results obtained from the simulations for the quadrature detector receiver. The minimum acceptable BER is $10^{-3}$ [22], therefore the reference $\frac{E_b}{N_o}$ is 15.75 dB, measured with $f_o = 0$ and $\alpha = 0$. This agrees with the performance of the quadrature detector with bit slicing decision algorithm reported in [25] and also with the results in [43][44][46].

The quadrature detector clearly shows a marked deterioration of performance as $f_o$ increases. With an $f_o$ of 70 kHz an SNR of 30 dB was required to achieve a BER of $10^{-3}$, an additional 14.25 dB compared to the reference level. When carrier frequency drifts were present in the transmitted signal a further degradation in performance was noted to the extent of 4.5 dB for the maximum drift rate.

### 5.2.2 Phase Shift Discriminator

Figures 5.5, 5.6 and 5.7 show the results obtained from the phase shift discriminator simulations. They are comparable to those obtained from the quadrature detector, indeed the degradation in performance due to a carrier offset of 70 kHz was the same. Also, the baseline performance figure of 15.75 dB was the same, which contrasts to the 1.5 dB advantage over the quadrature detector described in [25]. This is likely due to the quadrature detector performing slightly better in these simulations because of the higher sampling frequency of 320 MHz as opposed to 80 MHz. As with the quadrature detector, introducing a carrier drift into the transmitted signal caused a degradation in performance of 4.5 dB.

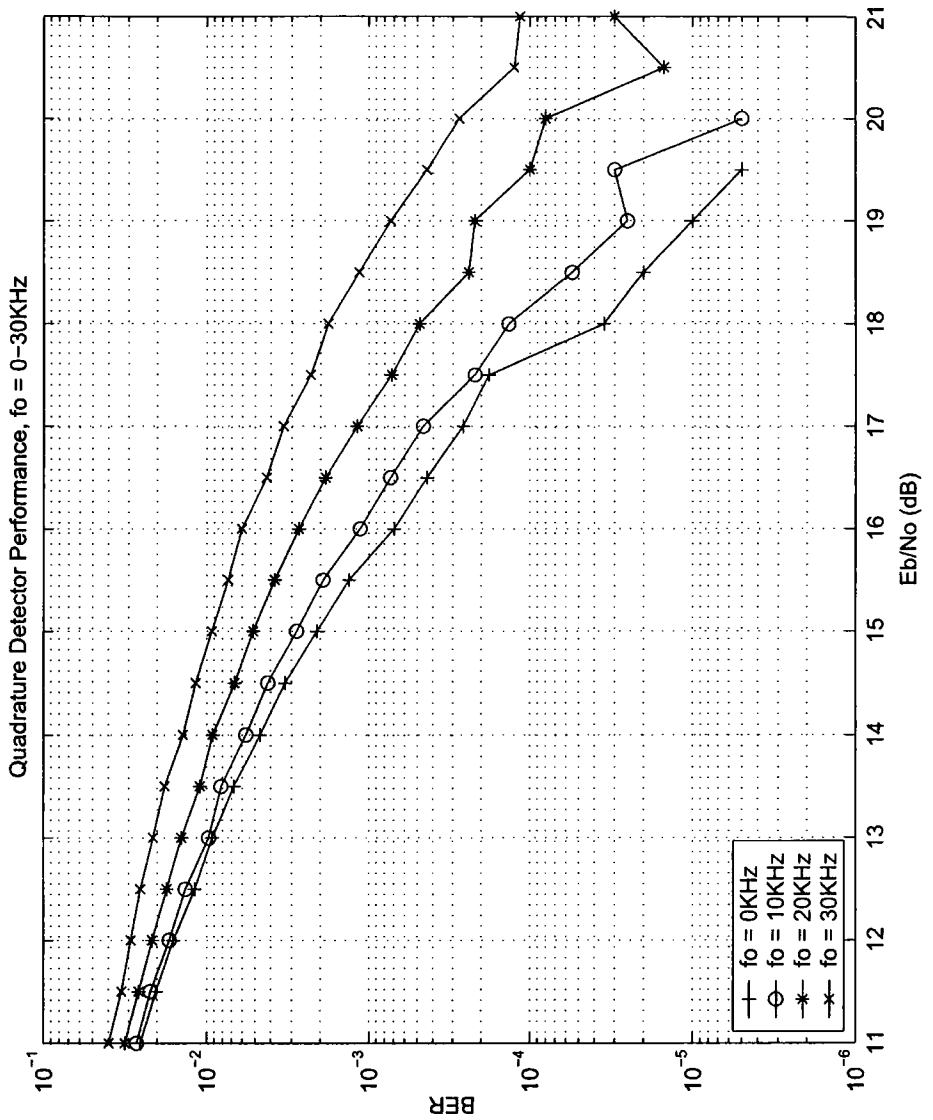


Figure 5.2: Quadrature Detector Performance: $f_o = 0\text{kHz} - 30\text{kHz}$

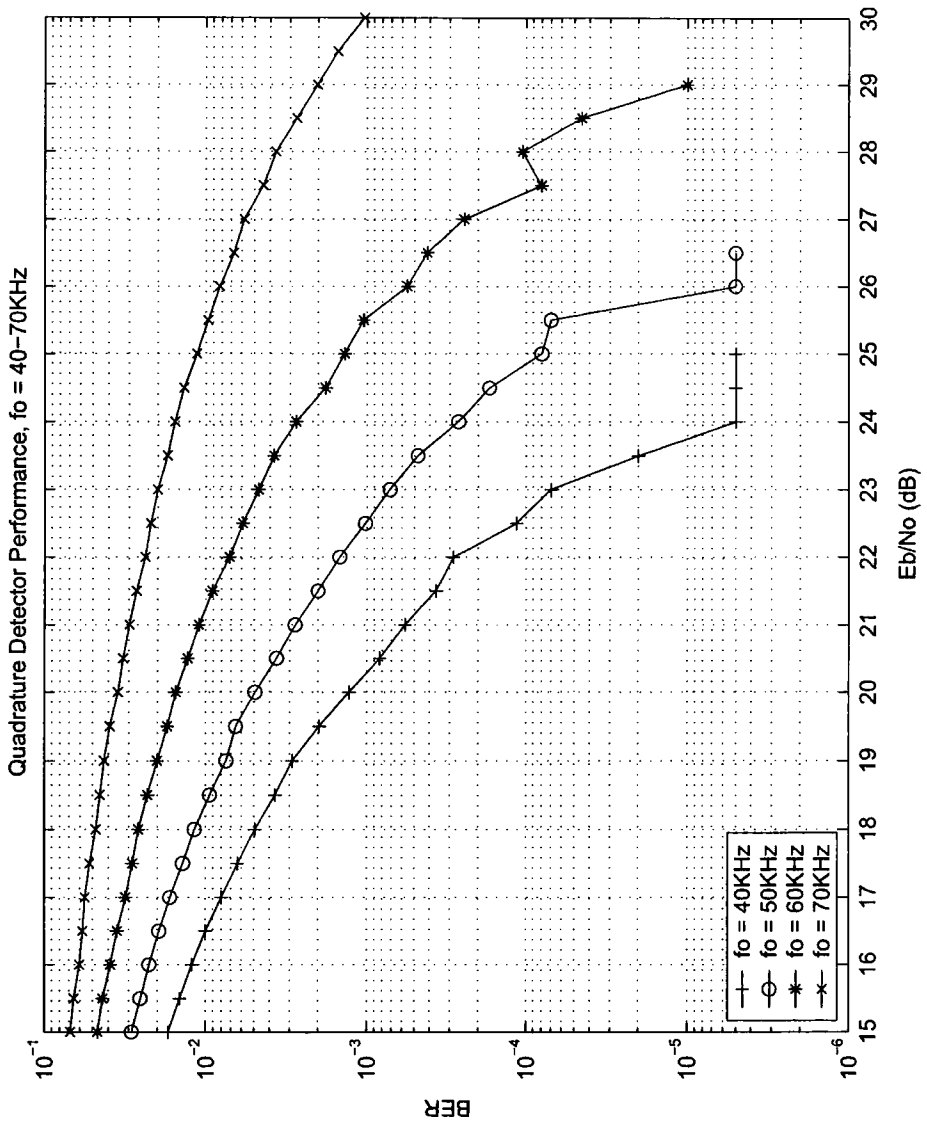


Figure 5.3: Quadrature Detector Performance: $f_o = 40\text{kHz} - 70\text{kHz}$

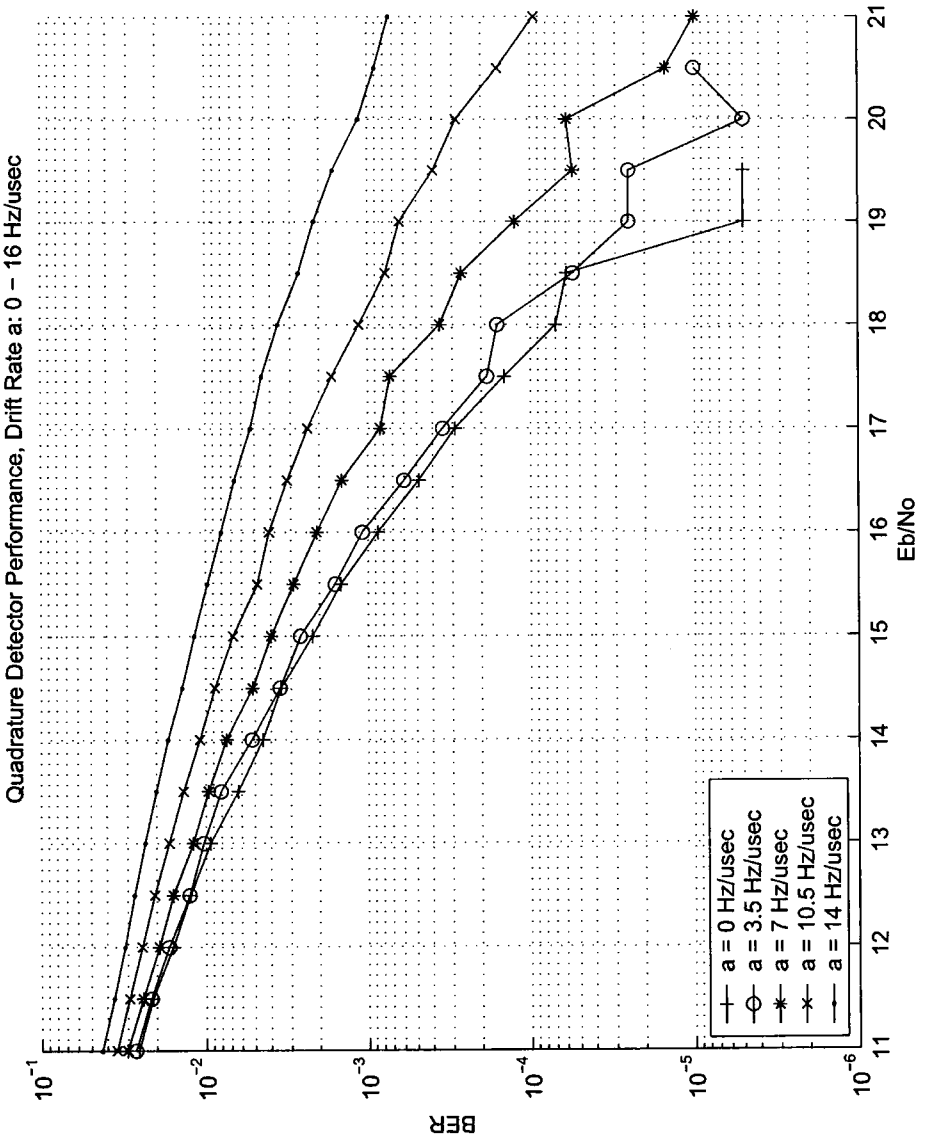


Figure 5.4: Quadrature Detector Performance: $\alpha = 0\text{Hz}\mu\text{s}^{-1} - 14\text{Hz}\mu\text{s}^{-1}$

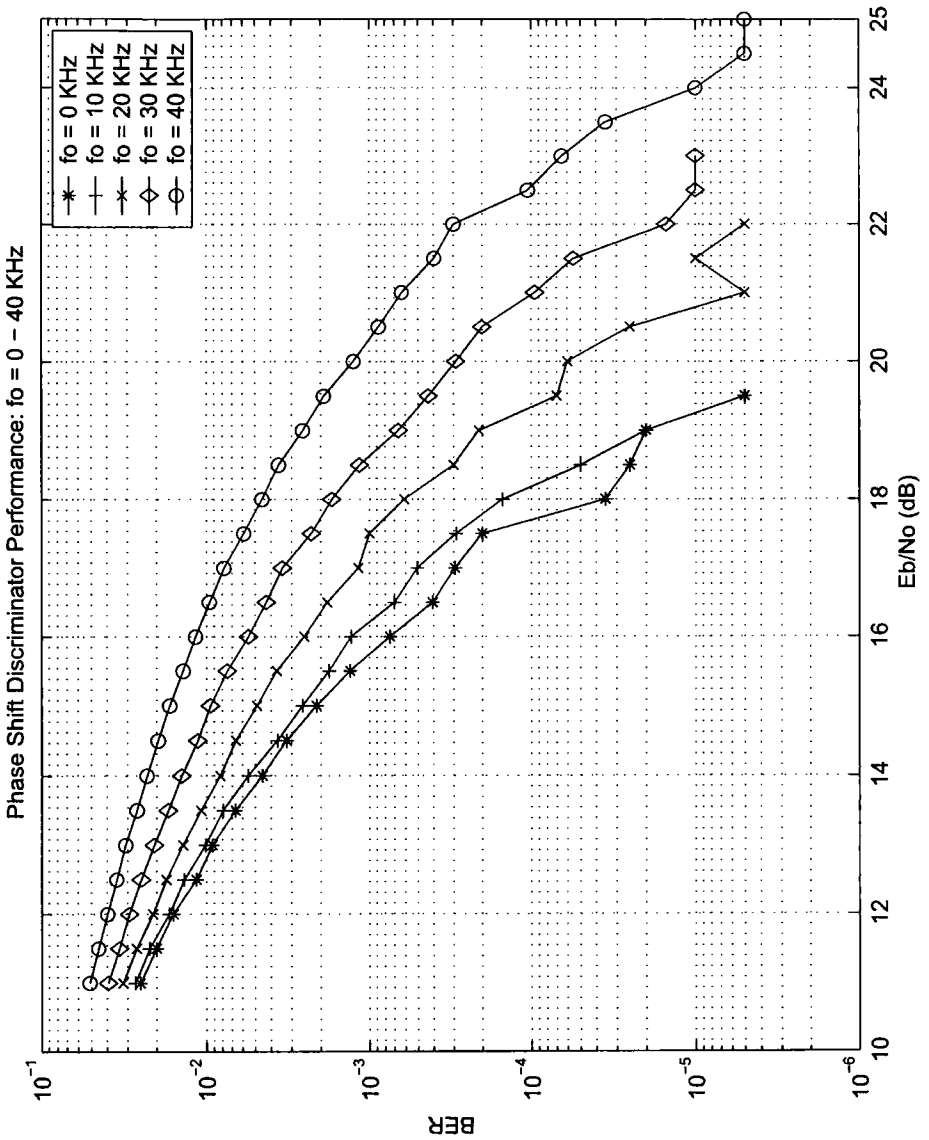


Figure 5.5: Phase Shift Discriminator Performance: $f_o = 0\text{kHz} - 40\text{kHz}$

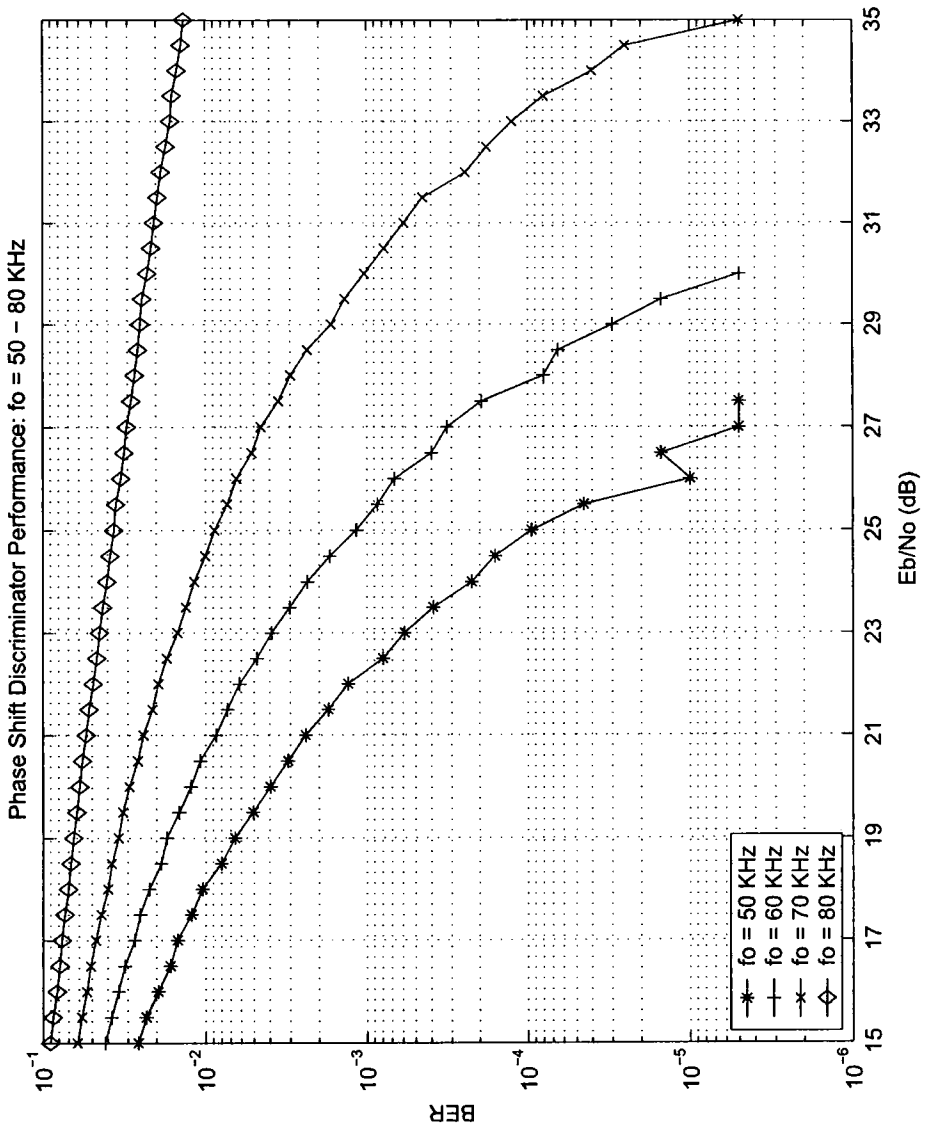


Figure 5.6: Phase Shift Discriminator Performance: $f_o = 50\text{kHz} - 80\text{kHz}$

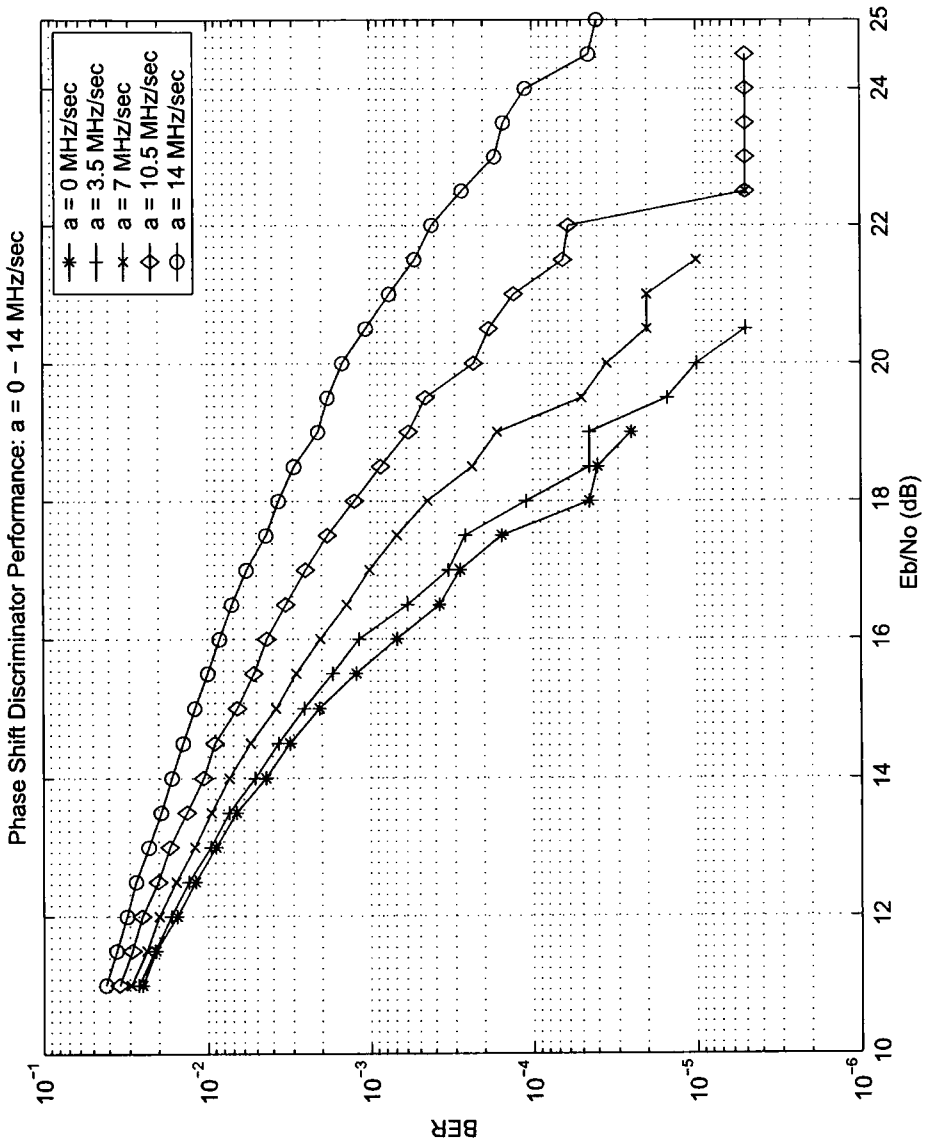


Figure 5.7: Phase Shift Discriminator Performance: $\alpha = 0\text{MHzs}^{-1} - 14\text{MHzs}^{-1}$

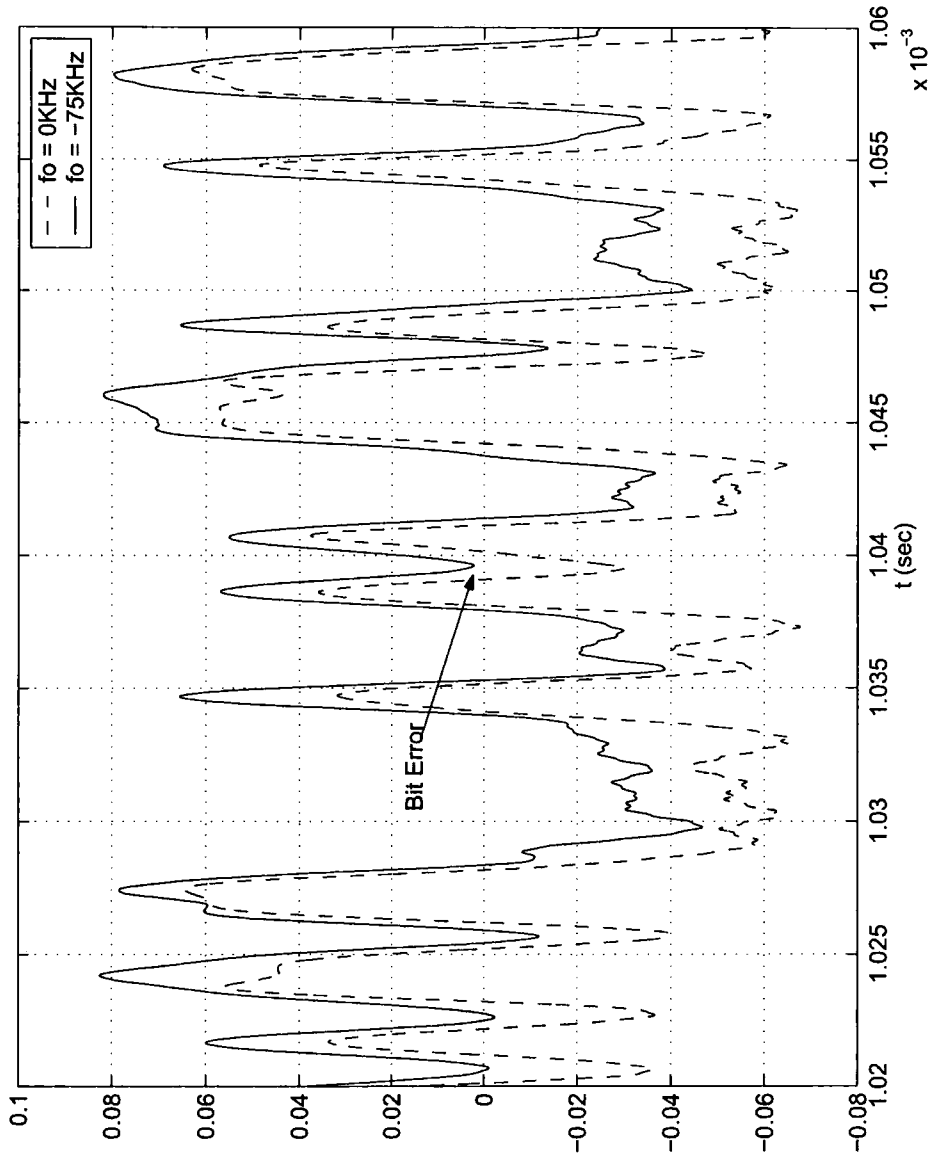


Figure 5.8: Quadrature Detector Output, $E_b/N_o = 20$dB

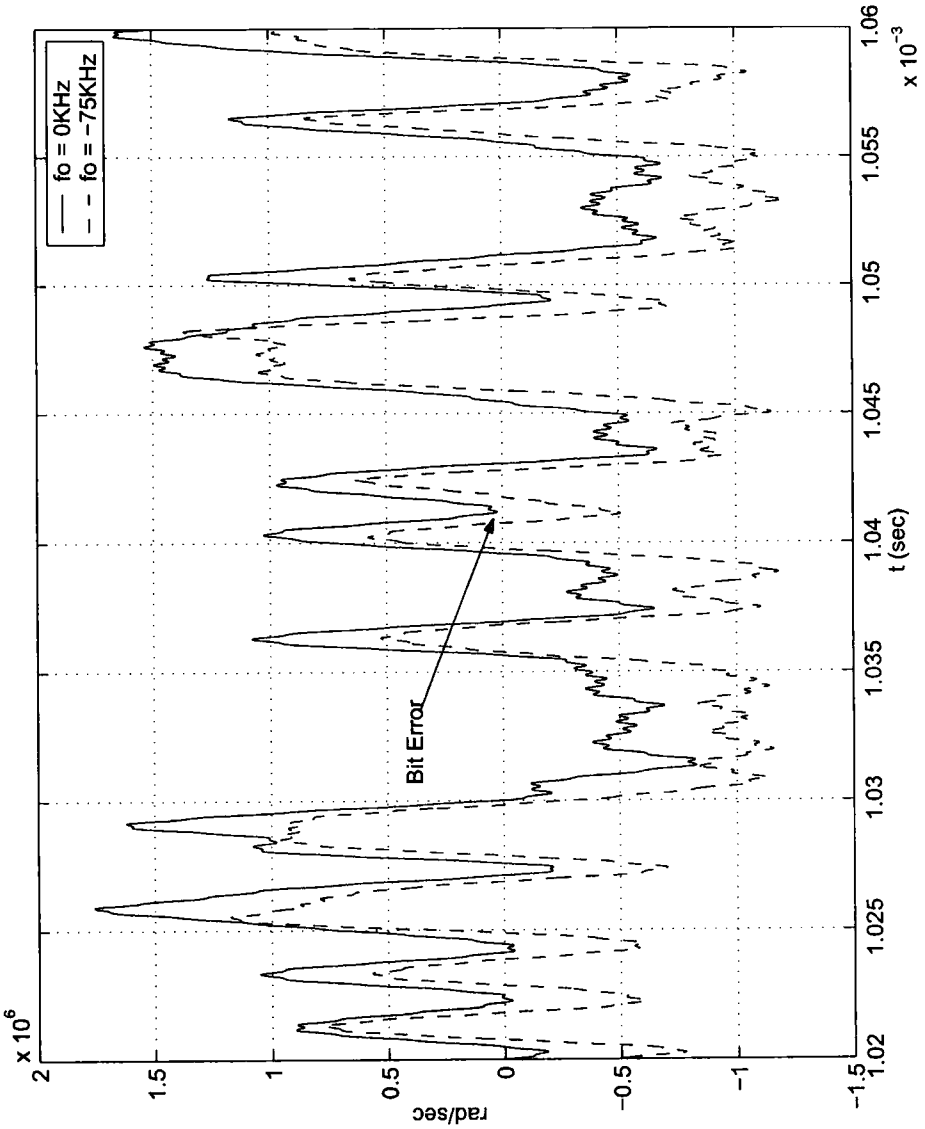


Figure 5.9: Phase Shift Discriminator Output, $E_b/N_o = 20$dB

## 5.3 Conclusions

These simulations have shown that significant degradations in receiver performance can be caused by transmitted frequency errors in Bluetooth systems. The consequences of such errors would ultimately be an apparently random and inexplicable reduction in throughput due to dropped packet retransmission, or a noticeable drop in link quality for asynchronous links due to packets being lost altogether. In the extreme an unfortunate coincidence of transmitter and receiver frequency offsets could cause a link to be lost completely. A point to be stressed here is that Bluetooth has no explicit receiver carrier frequency tolerance specification. There is a loosely implied requirement as in most implementations, if not all, the receiver LO will be the same as the free running transmitter LO (non-coherent) and will therefore conform to the same specification. However, this does have a serious consequence. Two compliant devices could display a mutual initial carrier offset of 150 kHz rising to, in the extreme, 230 kHz during a 3 or 5 slot packet. Figure 5.10 shows the effect of a 150 kHz mutual offset upon the output of both demodulation algorithms. In this extreme case the BER tends towards 0.375 as noise is reduced and is never greater than 0.3 at any SNR.

The effect predicted by the simulation results has been observed by engineers at CSR [49]. When characterising chips with communications analysers, the transmitted carrier frequency was found to be within tolerance but BERs were reported as out of specification, that is $> 10^{-3}$. These tests were in a low noise environment as the RF interface between DUT and tester was via a cable. The supposition is that the BERs were artificially high due to non-compliant or poor receivers in the test equipment failing to cope with a reasonable level of carrier errors in the manner demonstrated above.

The root cause of the elevated BERs is the misinterpretation of symbols in the bit slicer decision algorithm, due to a DC offset introduced into the demodulated NRZ signal. This can be seen if the phase of the received signal with $f_o \neq 0$ is

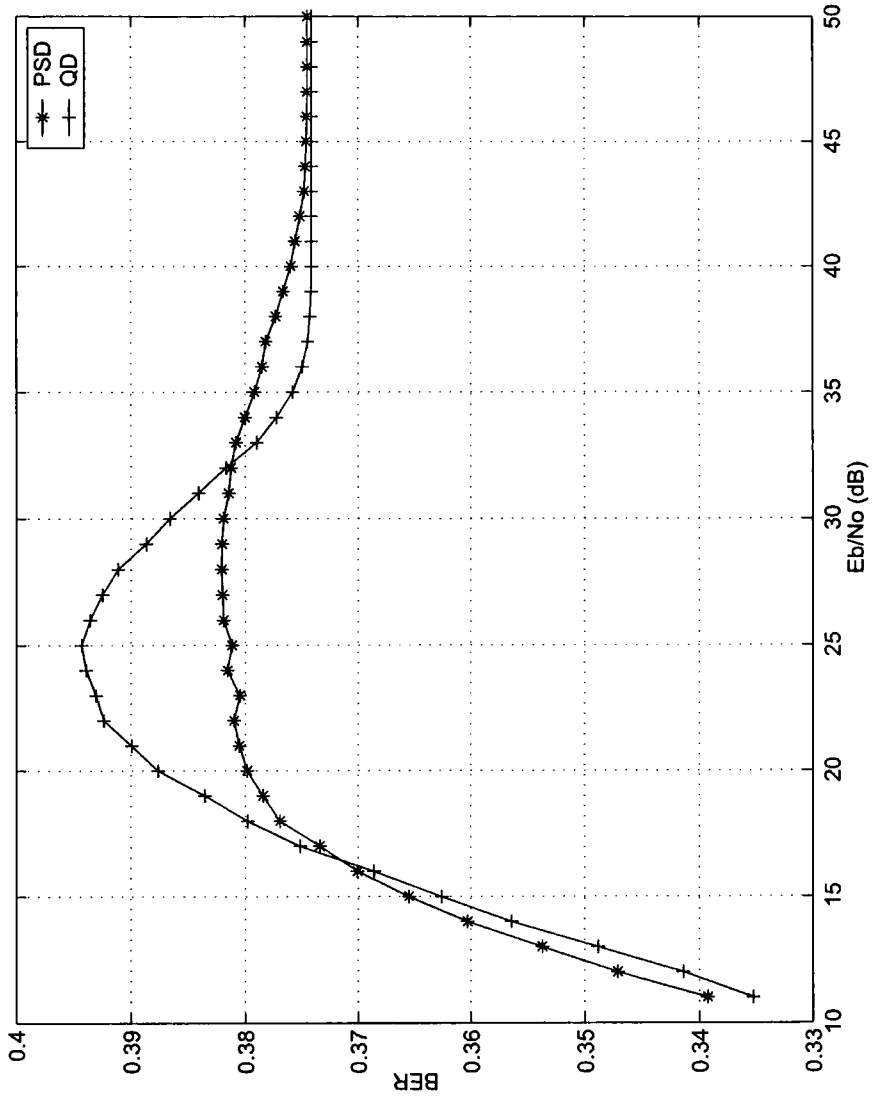


Figure 5.10: Demodulator Performance with $f_o = 150$kHz.

considered:

$$\phi(t) = -(2\pi f_o t + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau\ ). \tag{5.6}$$

After differentiation, as in the phase shift discriminator, this becomes:

$$\phi'(t) = -(2\pi f_o + 2\pi h m(t))\ . \tag{5.7}$$

Figures 5.8 and 5.9 illustrate this DC offset effect for an $f_o$ of -75 kHz. In figure 5.8 the quadrature detector output is shown for the same data modulated with and without a carrier offset, and with the same amount noise added. A bit which would be misinterpreted as a 1 by the bit slicer can be clearly seen. Figure 5.9 shows the output of the phase shift discriminator for the same data under the same conditions and the same bit can be seen to be in error. These graphs also serve to explain why comparable results were obtained from both demodulators: they fail in the same way at the same place.

Figures 5.8 and 5.9 also display an interesting feature of GFSK demodulation. A consequence of the pre-modulation filtering and the resultant ISI (see also figure 4.3), is the reduction in the peak amplitude of bits which are adjacent to two other bits of the opposite value ([...101...] for example). Since the final frequency deviation used to encode data in FSK is proportional to the modulating signal amplitude, this results in reduced deviations for alternating bits. This is the reason for the range of values allowed for modulation index (h) in the specification, the values are due to the Gaussian filter's parameters. It also explains the result shown in figure 5.10, where BER tended to 0.375. The probability of a [...101...] pattern for uniformly distributed bits is 0.375; all '0's in the packets were being incorrectly decoded. The deviation with which they were encoded, $\simeq$ 140kHz, is less than the combined frequency error of 150 kHz. Packets experiencing this level of frequency error could never be correctly received.

# Chapter 6

# Bluetooth Carrier Frequency Error Measurement

## 6.1 Motivation and Statement of Problem

In the previous chapter the detrimental effect of carrier frequency deviations upon GFSK receiver bit error rates was discussed. The results underline the importance of quantifying the modulation characteristics of a Bluetooth transmitter to ensure good link performance. To that end, this chapter will describe a new method of extracting the transmitter modulation characteristics, as defined by the Bluetooth specification, from sampled Bluetooth packet data. The method is based upon the Phase Shift Discriminator demodulator (section 4.2.1.2) and was implemented and tested upon an automated RF IC tester from the Japanese ATE manufacturer Advantest, at their European headquarters in Munich, Germany.

The Bluetooth specification [22] both defines the allowed range of transmitter modulation characteristics (section 3.3) and requires that they are tested during qualification and manufacture. To briefly reiterate this specification, the three parameters are: initial carrier frequency offset within 75 kHz of the nominal value; carrier drift less than 25 kHz or 40 kHz, depending upon packet length; and a

maximum instantaneous drift which must be less than 400 $Hz\mu s^{-1}$. By definition then, the initial carrier offset requires the measurement of the transmitted carrier frequency at the beginning of a packet. Also it is convenient to measure the carrier drift at the end of a packet. In practice this means that measurements must be made during the four bit preamble, that is in the first 4 μs of a packet. For consistency the term 'post-amble' is introduced here, and used to refer to an alternating four bit pattern in the last 4 μs of a packet. Carrier frequency drifts will be measured during the post-amble. Commercial products, such as Rohde and Schwarz's CMU200 Universal Radio Communications Tester [50] and the Anritsu MT8850A Bluetooth Test Set [51], will measure these frequencies to within 2 kHz and 1 kHz respectively. To be useful, therefore, the method described below must meet or exceed this resolution.

A method of measuring carrier errors independently of dedicated Bluetooth testers or communications anlysers is desirable in the context of ATE for RF ICs. Communications analysers are ideal for qualification testing, module test and final product test when Bluetooth ICs have been integrated into end products such as mobile phones. However, there is also a requirement to test individual ICs as they are manufactured, using production ATE such as Teredyne's Catalyst μWave6000, Agilent's 93000 series, Credence's ASL 3000RF or RFx systems and Advantest's T7611 RF IC tester. The choice of ATE depends upon the chip being tested. If it is a single chip solution, such as CSR's BlueCore series, then a SoC tester like the Catalyst would be appropriate. If, on the other hand, the chip to be tested is a transceiver IC such as the MC13180 from Motorola then a dedicated RF IC test system like the T7611 would be preferable as its lower overall test cost reflects the reduced complexity of the DUT.

In either case, test cost is an important consideration. A popular way of reducing test cost by increasing test throughput is to test multiple devices at once. Simultaneous testing of multiple devices with communications analysers would imply that several analysers would be required per test system. This would be impractical

due to their physical size and cost[1]. Instead, parallel test sites are facilitated by providing wide bandwidth RF ports for analysis and generation of RF stimulus to DUTs. For instance, the T7611 has 16 RF ports which may be connected internally to analog down conversion electronics and analog to digital converters. Measurement of device parameters is then performed with DSP on one of the two embedded Sun SPARC processors. In this way, a wide variety of devices can be tested from Bluetooth transceivers to mixer ICs for the pure RF testers and single chip Bluetooth solutions to media processors for the more sophisticated SoC testers. Clearly for this strategy to work there must be DSP algorithms available on the ATE systems to process the captured signals and extract the required device performance parameters. The method described in this chapter is such an algorithm.

### 6.1.1 Measurement Problem

Conventional dynamic[2] signal analysis in mixed signal and analog ATE is mainly performed through Fourier transformations of captured time domain signals. There is no question that the transmitter performance measurements we are considering here are dynamic so one would logically turn to fourier analysis for a measurement solution. However, as stated above, the measurements of carrier frequency characteristics must be made in the first and last four microseconds of a Bluetooth packet. An FFT performed on the modulated preamble would return coefficients spaced at 250 kHz intervals from $-\frac{f_s}{2}$ to $\frac{f_s}{2}$, giving a total of $\frac{250\times10^3}{f_s}$ frequency bins. The width of the bins $\Delta f$ is determined by $T_w$, the length of the truncation window applied to the signal, in this case 4 µs and is given by [52]:

$$\Delta f = \frac{f_s}{N} = \frac{1}{T_w} = \frac{1}{4 \times 10^{-6}} = 250\text{kHz}, \tag{6.1}$$

[1]Interface and housing considerations aside, adding four Rohde and Schwarz CMU200 to the T7611 would increase the capital cost by around 25%.

[2]Dynamic is used to distinguish between DC, or static, device characteristics such as differential non-linearity (DNL) in ADCs, and AC, or dynamic, properties such as total harmonic distortion (THD).

where N is the number of samples. This is the fundamental limit on the resolution of an FFT of the preamble. However, in order to correctly determine the carrier frequency errors, an ability to distinguish between frequency differences on the order of 1 kHz is required. Therefore a simple FFT is not suitable.

There are ways of increasing the number of frequency bins in the FFT without sampling a signal for longer. The simplest of these is padding the sampled data with zeroes to effectively increase the data length and take an FFT of the padded data. Whilst this does provide more samples in the output, the result does not have a higher resolution but rather the additional samples are interpolated from the original, "lower" resolution, FFT. Hence no new information is revealed in the spectrum as no new information about the signal was inserted into the time domain data by the padding.

A more complicated approach is to use the Chirp $z$ Transform (CZT) [53] which allows the calculation of a DFT across some arbitrary range of frequencies within the Nyquist limit and with an arbitrary resolution. The basis of the CZT is the observation that the DFT $X(k)$ of an N point sequence $x(n)$ is equivalent to its $z$ transform $X(z)$ evaluated at N equally spaced points around the unit circle in the z-plane:

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n} \quad 0 \leq n \leq N-1 \tag{6.2}$$

$$\begin{aligned} X(k) &= \sum_{k=0}^{N-1} x(n) \exp\left(\frac{-j2\pi kn}{N}\right) \quad 0 \leq k \leq N-1 \\ &= X(z)|_{z_k = \exp\left(\frac{j2\pi k}{N}\right)} \end{aligned} \tag{6.3}$$

The CZT uses a general contour $z_k$:

$$z_k = AW^{-k} \quad k = 0, 1, \ldots M-1 \tag{6.4}$$

$$A = A_0 \exp(-j2\pi\theta_0) \tag{6.5}$$

$$W = W_0 \exp(-j2\pi\varphi_0), \tag{6.6}$$

where $A$ defines the starting point of the contour and $W^{M-1}$ the end point relative to $A$. If $A_0$ and $W_0$ are both 1 then the contour is an arc of the unit circle, that is a portion of the DFT. By choosing appropriate values for A, W and M it is possible to extract the spectrum of a sequence with arbitrary resolution and start and end frequencies. Therefore, the captured preamble data from a Bluetooth packet could be analysed to result in a spectrum centred on a nominal channel frequency with a resolution of 1 kHz. However, the effect of such an analysis would be to zoom in on a portion of the original DFT; the same could have been achieved by padding with an appropriate number of zeroes and discarding the samples of the DFT outside the frequency range of interest.

This discussion of fourier transform methods illuminates the problem of measuring carrier frequency errors in Bluetooth packets: there is not sufficient information in the preamble to use the usual transform methods. The frequency errors must therefore be found by some other means.

Fourier transform based methods are sometimes refereed to as non-parametric since they estimate the spectrum of a signal directly from the signal itself. Parametric methods, by contrast, attempt to estimate the parameters of a system which would generate the signal given an input of white noise. The spectrum of the signal is then taken to be the frequency response of the estimated system. Types of parametric methods are Moving Average (MA), Auto-Regressive (AR) and a combination Auto-Regressive Moving Average (ARMA) [54]. The systems being modelled are filters which reproduce an estimate of the time domain signal being analysed

from an input of white noise. The filter which estimates the signal is removing frequencies from the input leaving only those present in the signal itself. Hence the filter's frequency response is an estimate of the signal's spectrum. The parameters which are being estimated in these methods are the filter coefficients. In AR models the coefficients are those of an IIR filter whilst MA methods use FIR filters and ARMA models use a combination of both. There are a number of ways calculating the coefficients including the Yule-Walker and Burg algorithms for AR and ARMA approaches respectively.

Non-parametric methods are better than fourier transforms when there is a short length of data and additional resolution is required [55]. Another class of spectral estimation methods are so-called 'sub-space' techniques such as Multiple Signal Classification (MUSIC) and eigenvector methods. These are very good at detecting sinusoids in noise with low SNR. However, the remainder of this chapter goes on to describe an alternative solution which is based upon extracting frequency errors after demodulation of the packets by exploiting the particular structure of the GFSK signal.

## 6.2 Chosen Algorithm

In chapter 5 the general equation for a GFSK signal was modified by the addition of two terms to describe carrier frequency offsets and drifts. The resulting equation (5.1) was:

$$s(t) = \cos(2\pi f_c t + 2\pi f_o t + 2\pi\alpha(t)t^2 + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau). \tag{6.7}$$

By exploiting the effect of these two extra terms for $f_o$ and $\alpha$ upon the output of the Phase Shift Discriminator (PSD), it is possible to extract the carrier frequency errors as required.

Applying 6.7 to the PSD results in a phase signal after the arctan operation of

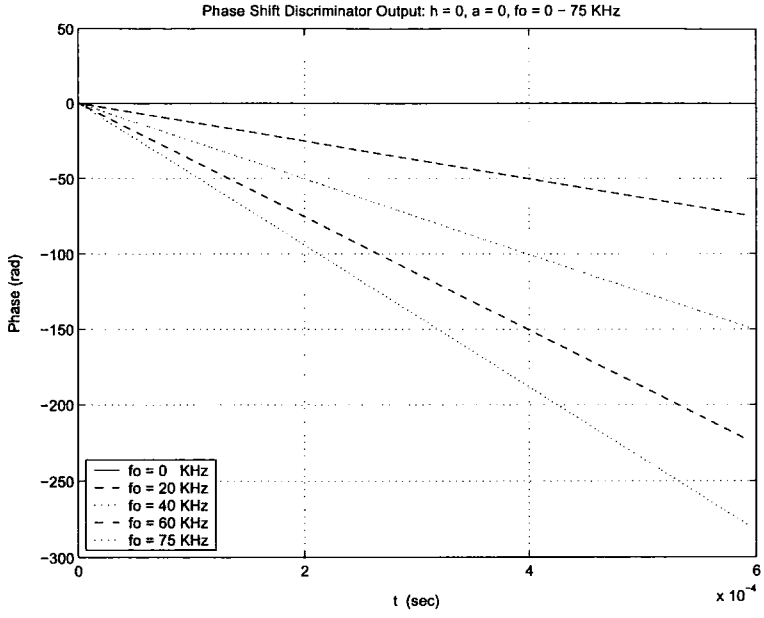


Figure 6.1: Phase Error: DH1 Packet, $f_o = 0\text{kHz} - 75\text{kHz}$

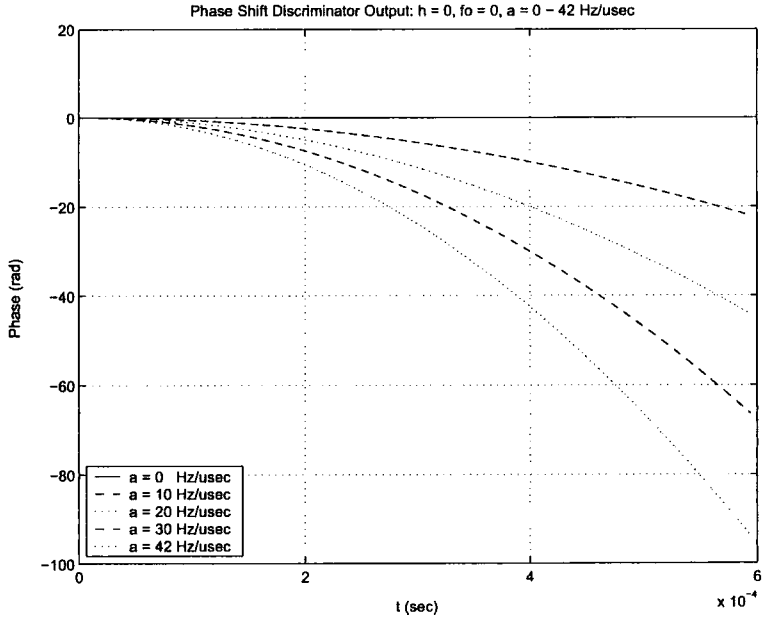


Figure 6.2: Phase Error: DH1 Packet, $\alpha = 0\text{Hz}\mu\text{s}^{-1} - 42\text{Hz}\mu\text{s}^{-1}$

the form:

$$\phi(t) = -(2\pi f_o t + 2\pi\alpha(t)t^2 + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau + \varphi_n(t)), \tag{6.8}$$

where $\varphi_n(t)$ is the angular component of applied channel noise. The carrier error information is within this phase signal: figure 6.1 shows $\phi(t)$ for varying $f_o$ whilst figure 6.2 shows the quadratic phase trajectory associated with a linear frequency chirp, that is constant $\alpha$. In both figures $h$ has been set to zero to remove the effect of $m(t)$ and no noise was added.

If $e'(t) = 2\pi h m(t) + \varphi_n'(t)$ then the differentiated phase $\phi'(t)$ can be expressed as:

$$\phi'(t) = -\left[2\pi f_o + \frac{\mathrm{d}(2\pi\alpha(t)t^2)}{\mathrm{d}t} + e'(t)\right], \tag{6.9}$$

and the instantaneous frequency error $f_e(t)$, which is the combined effect of $f_o$ and $\alpha$, is given by:

$$f_e(t) = \frac{\phi'(t) + e'(t)}{2\pi}. \tag{6.10}$$

$\phi'(t)$ can then be used to isolate the components of $f_e$:

$$f_o = f_e(0) = \frac{\phi'(0) + e'(0)}{2\pi} \tag{6.11}$$

taking $t = 0$ to be the start of the packet and therefore:

$$\alpha(t) = \frac{\phi(t) + (\phi'(0) + e'(0))t + e(t)}{2\pi t^2}. \tag{6.12}$$

However, if it is assumed that $\alpha(t) = K$ with $K$ a constant rate of drift, then

$$f_{dK}(t) = \frac{\phi'(t) + e'(t) + 2\pi f_o}{4\pi t}. \tag{6.13}$$

This value of $\alpha$ $\mathrm{Hzs}^{-1}$ can be used to find the frequency drift across a packet by evaluating (6.13) at $t = t_p$, where $t_p$ corresponds to the end of a packet. Multiplying the result by $t_p$ gives an answer in Hz. The assumption of constant drift rate is valid

when calculating carrier frequency drift as by definition it is the net drift, that is the final frequency error after the initial error $f_o$ has been taken into account.

Evaluating (6.11) and (6.13) at $t = 0$ and $t = t_p$ respectively is not practical due to the influence of the $e'(t)$ term in(6.11) and (6.13). This influence can be mitigated by finding the DC offset in the preamble and postamble by taking the mean of the sample values for the sets of four symbols. The mean will be zero if there is no offset and non zero if there is one. This technique removes the effect of the $m(t)$ component in $e'(t)$ as the modulated 1s are balanced by an equal number of modulated 0s, assuming that the deviation caused by a 1 is the same, but opposite, to that produced by a 0. The noise component is still present. Using $t = 0$ is still valid for the preamble and $f_0$ but finding the mean sample value in the postamble requires that the $t_p$ be modified to be the average time of the postamble symbols, i.e. $t_p$ = (mean postamble symbol number) $\times$ 1$\mu$s.

## 6.3 Implementation and Test Platform

Implementation and testing was done in conjunction with Advantest, following an invitation to present the results of the research into Bluetooth receiver performance at their Open House customer event in Munich. This was timed to coincide with the 2003 Semicon Europa. Advantest's motivation for the presentation was to promote the capabilities of their T7611 RF IC test system whilst highlighting their cooperation with academia. Our joint technical goal was to demonstrate that the T7611 could support additional user libraries of signal analysis routines by being able to automatically measure Bluetooth carrier frequency errors, with the measurement method being fully integrated into the T7611's software environment. This provided an excellent opportunity to validate the algorithm on commercial ATE by collecting a large amount of results from the analysis of real signals.

### 6.3.1 Test Setup

In outline, the algorithm was tested by repeatedly generating DH1 type packets with carrier frequency errors in the ISM band and capturing them with an ADC so that they could be passed to the measurement algorithm software. Figure 6.3 illustrates this arrangement.

Data files for the arbitrary waveform generator (AWG) were written by a revision of the Matlab transmitter code used in the previous simulations, modified to use a sampling frequency of 40 MHz, the AWG's maximum sample rate. DH1 packets were used as the AWG's sample memory is 40 ksample deep, with each sample being 12 bits wide. This gives a maximum sampling window of 1 ms which a 590 bit DH1 packet will fit into comfortably. The AWG data was arranged so that the 590 μs of packet was in the centre of the 40 ksamples with the remaining space before and after padded with zeroes. The packet payloads were in the PRBS9 pattern as defined in the Bluetooth specification and generated by Rhode and Schwarz's SMIG-K5 software [56], with appropriate post and preambles added. The Matlab code modulated the packet onto an intermediate carrier frequency of 5MHz, with carrier drift rates $0\text{MHzs}^{-1}$, $21\text{MHzs}^{-1}$ and $42\text{MHzs}^{-1}$. This resulted in three separate data sets for programming the AWG[3].

The output of the AWG was up-converted to ISM band RF at 2.405 GHz using a LO frequency of 2.400 GHz (LO1 in figure 6.3). This signal was sent to an RF output port connected directly to an RF input port with 50 Ω cable. Down-conversion with $LO2 = 2.400\text{GHz}$ was then performed before the resulting 5 MHz IF waveform was digitised, again at 40 MHz by the T7611's ADC. Triggering of the AWG and ADC was synchronised so that the captured data corresponded closely in timing to the AWG input data. This does not have to be exact, however, as there is approximately 0.205 ms of noise before the packet proper is seen. Whilst carrier drift errors were embedded in the AWG data files, initial carrier frequency offsets

[3] $42\ \text{MHzs}^{-1}$ corresponds to the maximum constant drift rate for a single slot 590 μs packet to conform to the 20 kHz net drift specification.

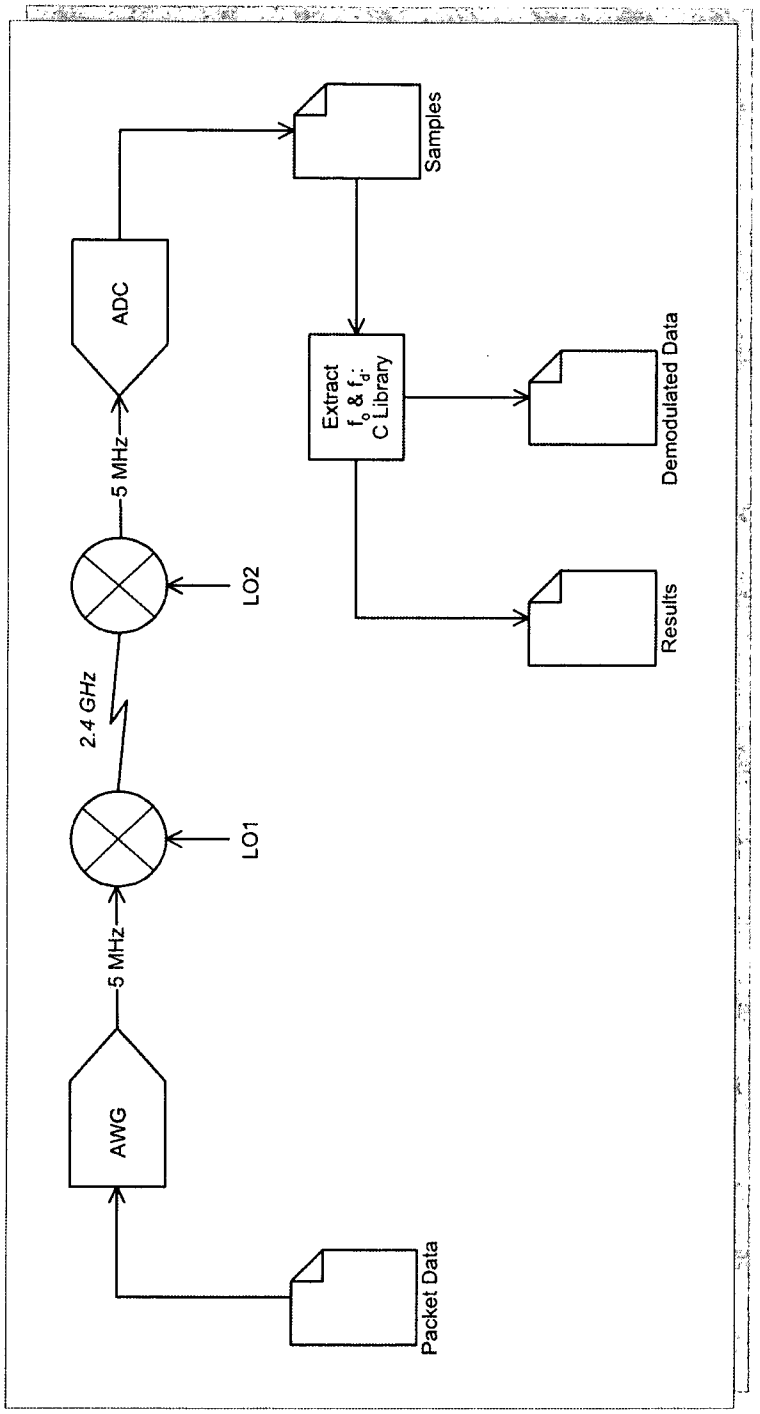


Figure 6.3: T7611 Test Setup

were introduced by small differences between LO1 and LO2. Three offsets were used 0kHz, 30kHz and 75 kHz. To give a transmitted offset of +75kHz LO1 was set to 2.40075 GHz with LO2 fixed at 2.40000 GHz. Both LOs were driven from a common reference oscillator and were calibrated. In total there were nine different combinations of carrier frequency error used, $0\text{MHzs}^{-1}$, $21\text{MHzs}^{-1}$ and $42\text{MHzs}^{-1}$ drift at each of the three initial offsets.

### 6.3.2 Software Implementation

The input to the measurement software is the 40k samples with a DH1 packet at an IF of 5 MHz embedded in them. Extracting the transmitted carrier errors from the packet data proceeds in three main steps. First the packet is found within the noise before and after it and the noise is discarded. Then the IF signal is demodulated by the phase shift discriminator method. Finally the samples pertaining to the packets pre-amble and post-amble are identified and processed to give $f_o$ and $\alpha$.

The packet signal is found within the noise before and after it to prevent "click noise" from obscuring the demodulated data. Click noise is a phenomenon in FM reception with high SNR. It is caused by the phasor sum of the signal and noise vectors rotating rapidly through $\pm 2\pi$ radians. When this rapid transition is differentiated, by a discriminator in analogue reception, the result is a spike in the demodulated signal which sounds like a click. At the start and end of the captured data the SNR is very high as the signal amplitude is zero. Therefore, passing all 40 k samples through the phase shift discriminator results in an output contaminated with large spikes either side of the correctly demodulated packet. These spikes are approximately ten times the peak to peak amplitude of the packet and so make automatic detection of the preamble and postamble much more difficult, if not impossible.

The extremities of the packet are found with a simple search algorithm. Figure 6.4 shows the start of a packet within the sample data. The search algorithm simply steps along the data until a sample with amplitude greater than a threshold

value of 0.1 is encountered. It then searches backwards from that point until a sample less than zero is found. The index of this sample is recorded and becomes the start of the packet. Finding the end of the packet works in the same way but in the opposite sense, as the search begins at the last sample and proceeds backwards. The noise is discarded by copying the packet data into an appropriately sized array. This approach has a further advantage in that it removes any reliance upon synchronisation of the AWG and ADC triggering. Furthermore, knowledge of the signal path delay between them is not required. Ease of portability of the measurement method between ATE platforms is therefore retained.

The next stage is quadrature down conversion of the 5 MHz IF signal and low pass filtering at baseband. Down conversion is achieved by multiplying the packet samples by $cos(2\pi f_c t)$ to give the in phase component and $sin(2\pi f_c t)$ for the quadrature component, where $f_c$ = 5MHz and $t = 0, 1/fs, \ldots, N/f_s$, with N being the number of samples remaining after the noise removal operation.

The low pass filtering is done by a 7 order Butterworth IIR filter with a 3 dB cut off frequency of 1 MHz. It was designed with the online tool by A. J. Fisher [57]. Each of the baseband signals is processed to remove the mixing products at 10 MHz.

The phase of the received signal is calculated with the four quadrant arctangent operator `arctan2` (part of the ISO9899 standard C math library). The resulting array of N values is constrained to $\pm\pi$. This is known as the "wrapped" phase and must be unwrapped to give the absolute phase. If the demodulation was to proceed without phase unwrapping then the subsequent differentiation would produce spikes similar to click noise in the output. The spikes here are caused by $2\pi$ discontinuities in the wrapped signal. An algorithm to unwrap a phase signal was first analysed by Itoh [58]. He concluded that a phase tracking procedure is a cascade of three simple operations: differentiating, wrapping and integration. This is equivalent to a sample by sample sum of the wrapped phase differences.

Since the IF signal was sampled at above the Nyquist rate of 10 MHz, the phase

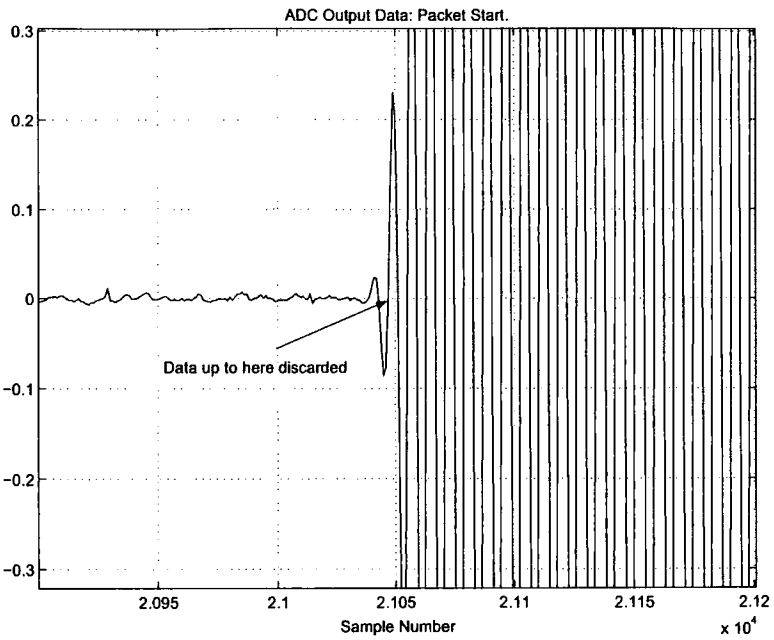


Figure 6.4: Start of a Packet in ADC Output Data

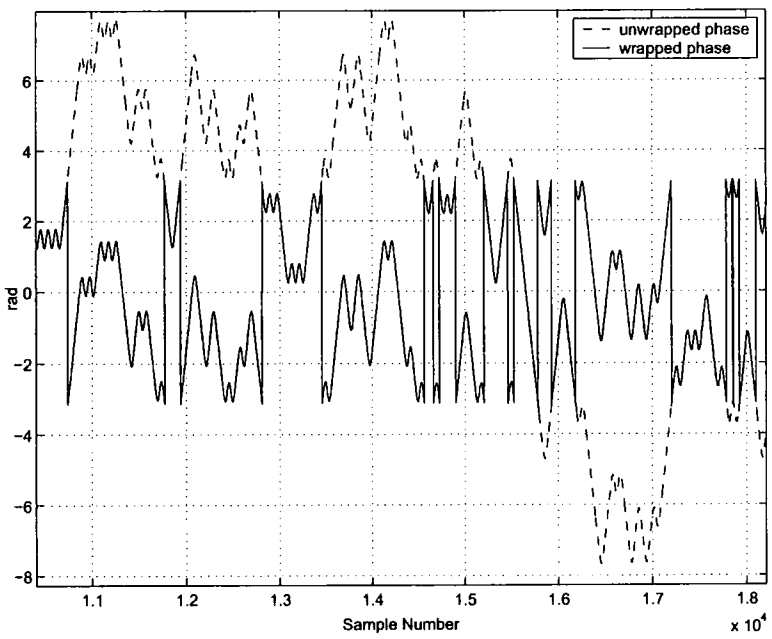


Figure 6.5: Wrapped and Absolute Phase Plots

can be recovered without aliasing. Nyquist sampling is equivalent to restricting the sample to sample phase difference to less than $\pm\pi$ everywhere [59]. Therefore, any phase differences greater in magnitude than $\pi$ are due to wrapping and their removal will result in the true absolute phase.

Figure 6.5 shows the wrapped and unwrapped phases from part of the packet shown in figure 6.4. The algorithm developed to unwrap the phase in figure 6.5 differs from Itoh's in that it finds discontinuities and adds or subtracts $2\pi$ from the samples between discontinuities as appropriate using a recursive function. Each pass through the function builds a linked list of structures in memory which describe the position and direction of discontinuities in the input phase signal. This list is then processed to remove the discontinuities by adding $2\pi$ between a negative going and a positive going step, for example between the first two discontinuities shown in figure 6.5. Between positive going and negative going steps $2\pi$ is subtracted. In order to ensure that all discontinuities are dealt with, the function calls itself recursively until the list of structures is empty. The lists are then removed from memory and the unwrapped phase is returned.

The final demodulation operation is the differentiation of the unwrapped phase. There are two possibilities: the simple arithmetic approach where $x'[n] = x[n] - x[n-1]$ or using an FIR filter with a frequency response $H(\omega) = j\omega$. The first method gives an output of the correct shape but trials showed that its amplitude was severely suppressed. Therefore, the filtering method was used. An appropriate FIR filter was designed using Matlab's `remez` function which implements the Remez exchange algorithm. The filter had 1001 taps, a large number but the choice is justified by the error profile of the design algorithm when compared to the ideal filter. The error is weighted by $1/f$ so that it is mostly constrained to the higher frequencies with the peak error magnitude increasing linearly as the normalised frequency approaches 0.5 ($f_s/2$) [53]. The rate of increase is inversely proportional to the length of the filter. Hence, using many taps minimises the demodulator

output error due to the differentiation. An improvement in measurement speed could be gained by using a shorter filter. However, as the aim of the software was to demonstrate the effectiveness and accuracy of the method, speed was felt to be a secondary consideration.

To extract the initial carrier frequency offset, the DC component of the demodulated preamble must be calculated. This is done by first finding the sample indices of the five zero crossing points in the preamble, see figure 6.6. A running sum of the sample values between the first and last of these indices is then made. The result is divided by the number of samples in the summation to give the mean value of the preamble $\overline{x}$, its DC offset. To obtain a value for $f_o$ equation (6.11) is then evaluated with $\phi'(0) = \overline{x}$ and the error term $e'(0)$ is set to zero. This is a consequence of integrating across the preamble symbols, assuming they are symmetrical about $\overline{x}$.

Finding $\alpha$ follows a similar procedure. The five zero crossing points are found by searching backwards from the end of the packet which is given by the known packet length in samples. The mean value of the postamble is found and equation (6.12) is evaluated using the previously calculated value of $f_0$. The average index of the samples used in the summation multiplied by the sampling interval gives $t_p$. This provides a value for the net frequency drift rate which can be compared to that introduced into the AWG packet data.

The measured values which correspond to the data in figures 6.6 and 6.7 were $f_o = -74.79600\text{kHz}$ and $\alpha = 41.48105\text{MHzs}^{-1}$, equivalent to a net carrier frequency drift of 24.41 kHz($= \alpha * 590\mu\text{s}$).

## 6.4 Measurement Results

Tests were run multiple times for each of the nine combinations of initial carrier offset and drift rate. A summary of the results over a total of 2451 iterations is presented in table 6.1.

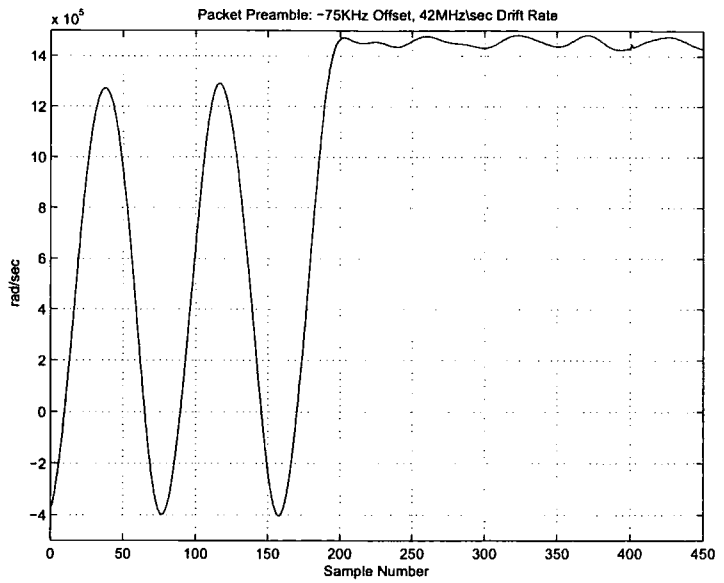


Figure 6.6: Demodulated Packet Preamble: -75 kHz Offset, 42 $MHzs^{-1}$ Drift Rate.

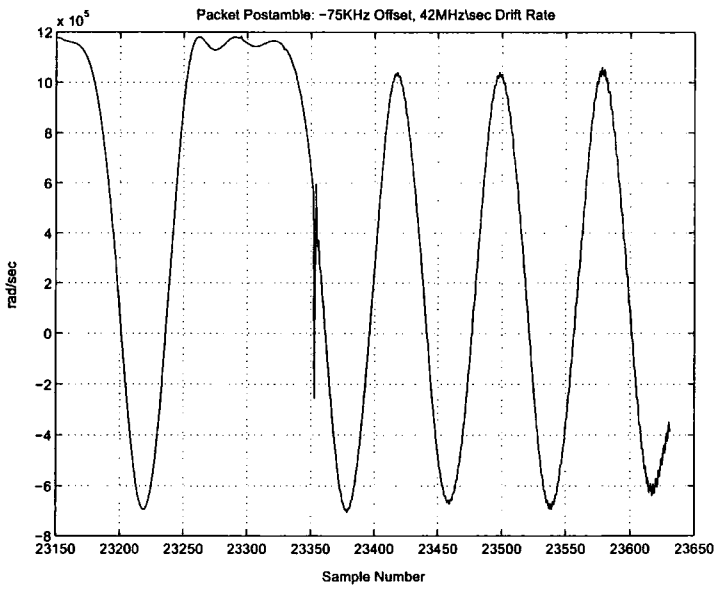


Figure 6.7: Demodulated Packet Postamble: -75 kHz Offset, 42 $MHzs^{-1}$ Drift Rate.

| Applied Offset | Applied Drift | n | Measured Offset (kHz) | | Measured Drift ($MHzs^{-1}$) | | Measured Net Drift (kHz) | |
|---|---|---|---|---|---|---|---|---|
| (kHz) | ($MHzs^{-1}$) | | $\bar{x}$ | $\sigma$ | $\bar{x}$ | $\sigma$ | $\bar{x}$ | $\sigma$ |
| 0 | 0 | 151 | 0.5739 | 0.3903 | 0.3723 | 0.4893 | 0.22 | 0.288 |
| 0 | 21 | 153 | 0.4178 | 0.4151 | 21.17 | 0.5206 | 12.46 | 0.306 |
| 0 | 42 | 301 | 0.2193 | 0.4292 | 41.89 | 0.5411 | 24.65 | 0.318 |
| 30 | 0 | 191 | 30.64 | 0.4521 | 0.5776 | 0.5988 | 0.34 | 0.352 |
| 30 | 21 | 301 | 30.49 | 0.4563 | 21.36 | 0.5161 | 12.57 | 0.304 |
| 30 | 42 | 301 | 30.23 | 0.4025 | 42.10 | 0.5073 | 24.78 | 0.299 |
| 75 | 0 | 301 | 75.46 | 0.4878 | -0.242 | 0.7863 | -0.14 | 0.463 |
| 75 | 21 | 301 | 75.25 | 0.4615 | 21.14 | 0.5655 | 12.44 | 0.333 |
| 75 | 42 | 451 | 75.06 | 0.4552 | 42.24 | 0.5473 | 24.86 | 0.322 |

Table 6.1: Summary of Results

The standard deviation of the measured initial offset frequencies was consistently less than 500 Hz with mean errors between 60 Hz and 574 Hz. It is therefore reasonable to quote an accuracy of 1 kHz for $f_o$ which is comparable to the Anritsu MT8850's performance [51]. Similarly, the measured drift rate accuracy is within 1 $\text{MHzs}^{-1}$, which translates to a net offset accuracy of less than 590 Hz for a DH1 packet.

Measurement errors were caused by: system noise; quantisation noise; differences in the nominal and actual local oscillator frequencies; and calculation errors in the measurement algorithm. The effect of noise is to alter the shape of the symbols, making them non symmetrical. This effect can be seen in the last two "0" symbols in figure 6.7. Since their symmetry is relied upon to remove the influence of $e'(t)$ in the calculations, then noise ultimately affects the measurement error. It is possible to mitigate for the modulated data when evaluating (6.11) and (6.12) but not the $n'(t)$ component of $e'(t)$. An approximate figure for the SNR in the sampled data is 47 dB, measured by comparing the peak to peak amplitude of the packet signal with that of the noise either side of it.

### 6.4.1 Assessment of Algorithm Performance

The method used to find the DC offsets in the preamble and post-amble was simple rectangular integration. In the absence of noise this would not contribute to measurement error as the integration errors from the two modulated ones would be offset exactly by those from the modulated zeroes, provided that the sampling of the four symbols were coherent. However, in general there was not an exact integer number of sampling instances between the five zero crossing points of the pre and post-ambles. This means that, even assuming perfect symmetry, there is some residual error due to the difference between the first and last samples in the summations. However, calculating this difference and removing it from the sum was not found to improve the measurement error. This leads to the conclusion that the error

was due to a combination of system noise and calculation errors in the integration. Therefore, the question which naturally arises is: to what extent is the standard deviation of the frequency estimates due to the captured signal's SNR?

To address this question it is necessary to know what standard deviation would be observed with a perfect estimation method. With such a method, a Maximum Likelihood Estimator (MLE), the standard deviation, $\sigma_f$, would be entirely due to the SNR of the sampled values and other *nuisance* parameters such as timing jitter in the ADC and phase noise. MLEs choose estimates of parameters, frequency offset in this case, which would have made the observed data most likely to occur [60]. Also, MLEs are said to be *statistically efficient* in that they produce estimates which have the lowest theoretically possible variance. Calculating this minimum variance, $\sigma_f^2$ , is done by evaluating the Cramer–Rao Lower Bound (CRLB). The CRLB on the standard deviation of the estimation error for the frequency of a single sinusoidal tone contaminated by AWGN is [61][62]:

$$CRLB(\sigma_f) = \frac{\sqrt{12}}{2\pi T\sqrt{(N(N^2-1)SNR)}}, \tag{6.14}$$

where $T$ is the sampling period (0.025 µs), $N$ is the number of samples from which the estimate is made and SNR is expressed as a power ratio not in dB[4]. To find an appropriate CRLB for these results, two values of $N$ could reasonably be selected: either $N = 160$, which corresponds to all four preamble symbols, or $N = 40$ corresponding to a single symbol period. Evaluating (6.14) with each value gives:

$$N = 40,\ \sigma_f = 389 \text{ Hz}$$

$$N = 160,\ \sigma_f = 48.7 \text{ Hz}.$$

However, since the measurement method described above is equivalent to finding the mean value of the instantaneous frequency at the centre of each of the four preamble

[4]i.e. for SNR = 47 dB, the correct value would be $10^{4.7} = 50119$.

symbols, the first result with $N = 40$ is the correct choice. Furthermore, since the ideal estimation process is identical for each of the four symbols (same SNR, T and N) the CRLB for four individual estimates of instantaneous frequency has the same value. Therefore, the mean value of the four CRLBs is equal to any one of them as the variance of the sum of independent random variables is equal to the sum of their variances [63]. Conformation of this comes from D'Andrea et al who derive a Modified CRLB (MCRLB) for the variance of frequency offset estimates ($\nu$) [64]:

$$MCRLB(\nu) = \frac{3}{2\pi^2 T^2 N^3 SNR}. \tag{6.15}$$

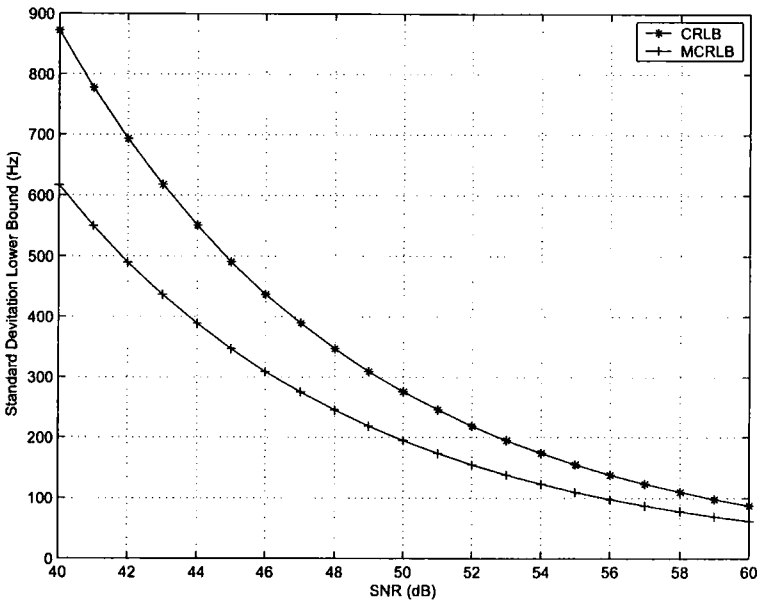


Figure 6.8: Dependance of CRLB and MCRLB on SNR ($N = 40$).

Evaluating (6.15) with the same values as before and $N = 40$ gives $MCRLB(\sigma_f) =$ 275.3 Hz. This is lower than the corresponding CRLB as the MCRLB is always a *looser*, that is smaller, bound [64]. The choice of $N = 40$ is therefore reasonable and the lowest possible $\sigma_f$ which could be achieved is 389 Hz.

Before proceeding it should be clarified that the CRLB gives a lower bound on the variance of the error $\hat{\lambda}(x) - \lambda$, where $\hat{\lambda}(x)$ is an estimator of $\lambda$. However, the

variance of a random variable $Y$ which is related to another random variable $X$ by $Y = c_1X + c_2$ is $\sigma_Y^2 = c_1^2\sigma_X^2$. Hence, if the error and the estimates are considered to be two random variables related by a constant which is the true value of the frequency offset (c.f. $c_2$), then their variances are equal. It is possible, therefore, to directly compare the lower bounds stated above with the standard deviations given in table 6.4.

The lowest standard deviation in initial frequency offset measurements achieved during the tests was 390 Hz with $f_o = 0$ Hz and $\alpha = 0$ MHzs$^{-1}$. The highest was 488 Hz with $f_o = 75$ Hz and $\alpha = 0$ MHzs$^{-1}$. The lower value is remarkably close to the CRLB of 389 Hz. This would seem to suggest that, in the case of no carrier frequency errors, the method almost achieves the most accurate result possible. However, such a conclusion could only be tentative as the measurements were made with only one level of SNR. In order to make a more conclusive statement, measurements would have to be made at several SNRs, with the expectation that the observed standard deviations would track the CRLB in the manner shown in figure 6.8. This graph illustrates the strong, inverse dependance of the CRLB on SNR. Small errors in the estimation of the SNR would lead to significant differences in the calculated lower bounds. For instance, if the true SNR with no frequency errors was 47.5 dB then the true CRLB would be 368 Hz. Unfortunately it is not possible to take measurements at several SNRs. A prudent conclusion would therefore be that the measurement method provides a good, but not statistically efficient, estimate of carrier frequency errors, with an estimation error distribution inversely proportional to signal SNR. Therefore, to answer the question posed above, errors produced by the method are largely due to the SNR of the digitised signal. Some improvement would be possible, perhaps through a more accurate integration method such as Simpson's. The maximum gain which could be achieved, on average, is around a 12 % reduction in standard deviation for $f_o$. However, since the method is not based upon a maximum likelihood evaluation, this level of improvement could not

practically be realised.

It can be stated that the measurement error for $f_o$ was $\leq$ 878Hz in 95.5 % of the measurements on the test system. Similarly the error for net frequency drift was $\leq$ 751 Hz. These values were found from the mean of the measurement's standard deviations and assumed that errors were normally distributed.

The Bluetooth RF Test Specification (version 1.1) describes how initial carrier frequency error measurements should be made in a general way. The method used here is broadly compatible with this description except in the data used to make calculations. The specifications states that:

> The measurement shall start at the center of the first preamble bit until the center of the first bit following the 4th preamble bit.

In this implementation the measurement starts at the beginning of the first preamble bit until the end of the fourth not the centre of the fifth. The reason for this can be seen by referring back to figure 6.6. Inclusion of the first half of the fifth bit (beyond about sample 175) would have the effect of over estimating the DC level of the preamble data and hence the magnitude of the frequency estimate. This is the case as the sixth bit in the packet is the same as the fifth so increasing magnitude at the centre of the fifth. The specification compensates for this effect by mandating that 10 packets are measured, although this is not explicitly stated.

## 6.5 Conclusions

In this chapter an algorithm which extracted carrier frequency error parameters, from Bluetooth packets captured at a low IF, was described. It was suited to software implementation in mixed signal or RF IC ATE systems. This is significant as a goal of ATE manufacturers is flexibility in the types of devices that their equipment is able to test. Being able to call upon a library of parameter estimation routines is therefore desirable. For these reasons, Advantest was keen to prove to its customers

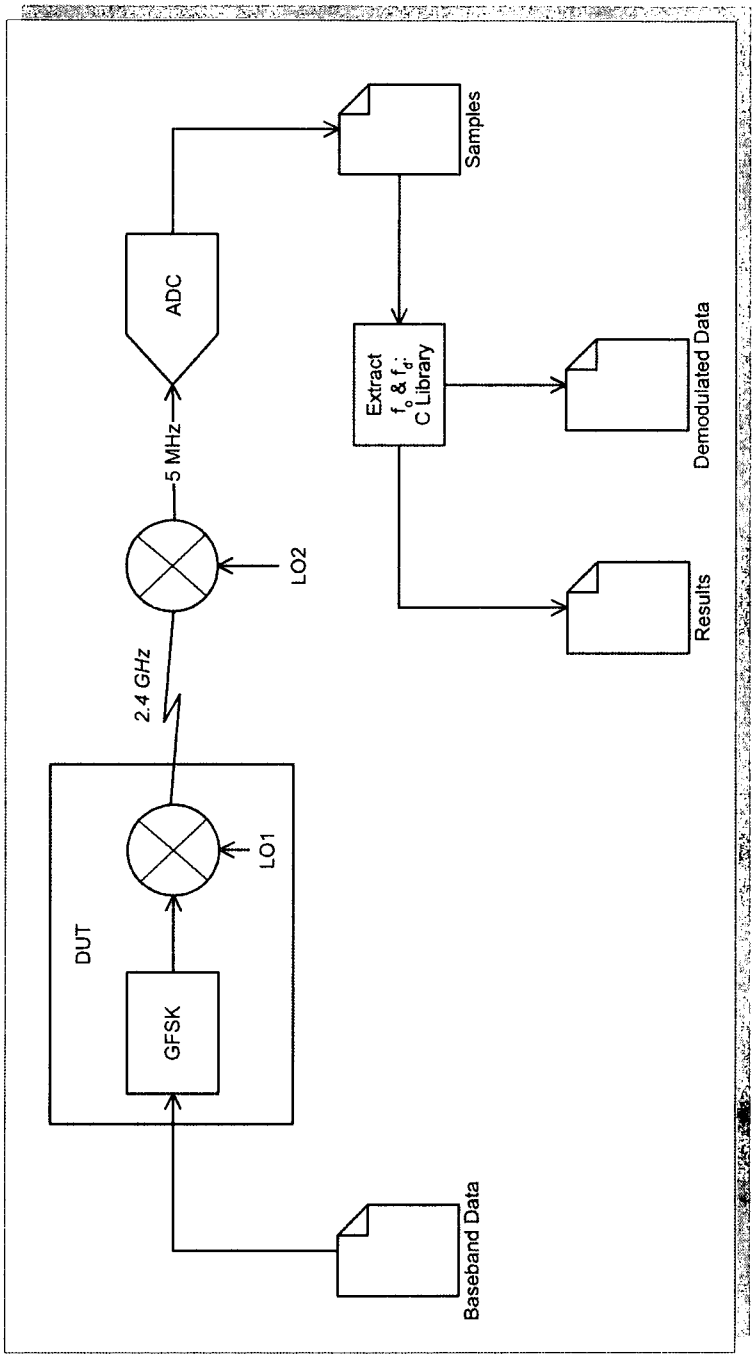


Figure 6.9: Transmitter DUT Test Setup.

that the T7611 RF IC test system offered such flexibility. In order to do so the carrier frequency error estimation method was implemented on the T7611 and demonstrated to Infineon and ST Microelectronics test engineers in Munich. Figure 6.9 shows a possible set up for production transmitter tests on a Bluetooth transceiver IC. It uses only the built in capabilities of the T7611 and the additional library routines for extraction of the transmitted carrier frequency characteristics as required by the Bluetooth specification.

A result of the demonstrations was a large set of test data, enabling an objective evaluation of the algorithm's performance. Conclusions were that whilst it did not provide estimates with standard deviations as low as theoretically possible given the information available, it did nevertheless give sufficiently good estimates: to within 900 Hz for initial carrier frequency offsets and within 800 Hz for net carrier frequency drifts, at an SNR of 47 dB.

# Chapter 7

# An Adaptive Threshold Decision Algorithm

## 7.1 Compensation for Carrier Frequency Errors

A relationship between received BERs and carrier frequency error characteristics has been established. It is clear that Bluetooth devices will experience degradations in performance if transmitted carrier frequency deviations from the ideal are not compensated for. Therefore, in this chapter a method of compensation for carrier deviations will be described and analysed through simulations.

When considering how to structure a receiver which is tolerant to transmitted carrier frequency errors, the first question which naturally arises is exactly what type of errors are to be compensated for? In chapter 5 it was shown that the largest effect on received BERs was caused by initial carrier offsets. In the worst case of a 150 kHz difference between transmitter and receiver carrier frequencies, the BER was $\gg 10^{-3}$ at all SNR. As a minimum, therefore, a compensation method must take initial carrier offsets into account. This specific issue will be addressed in this chapter.

### 7.1.1 Potential Compensation Methods

Ultimately, elevated BERs are the result of a spurious DC component in the demodulated signal which causes misinterpretation of symbols by the decision algorithm. Therefore, to compensate, either the way in which the decision algorithm operates must be changed or the DC offset must be constrained. In either case, an estimate of the magnitude of the carrier frequency error, $f_o$, is required. This can be obtained using the proven method described in chapter 6, leaving the question of how to use this information to effect a frequency correction. One way to approach this is to split the reception process into its constituent operations and consider where frequency errors could be mitigated for. Referring back to figure 4.4, there are three stages in a GFSK receiver: down-conversion from RF, demodulation and a symbol by symbol decision algorithm.

Compensation in the first stage implies adjustment of the local oscillator. However, it is steady state errors in the LOs which are responsible for the original frequency offset. Adapting the LO design to account for a correction factor, by whichever method, is likely to be counterproductive in terms of increased circuit complexity and cost. Furthermore, to be effective then the correction would need to be made between the end of the preamble and the beginning of the packet header, which are adjacent. The timing window would therefore be of the order of several hundred nanoseconds, very short in comparison to the inter-slot (frequency hop) tuning time of approximately 150 μs. These considerations inevitably lead to the conclusion that compensating for errors by adjusting the down-conversion LO is impractical.

Whilst modifying the RF to IF LO may be impractical, a similar strategy is to include an additional mixing stage after down-conversion, (figure 7.1), as suggested in [65] for residual frequency offset correction in OFDM receivers. However, as the signal bandwidth is greater than the sideband separation, the mixer used to facilitate the frequency corrections needs to be a balanced modulator design as is sometimes

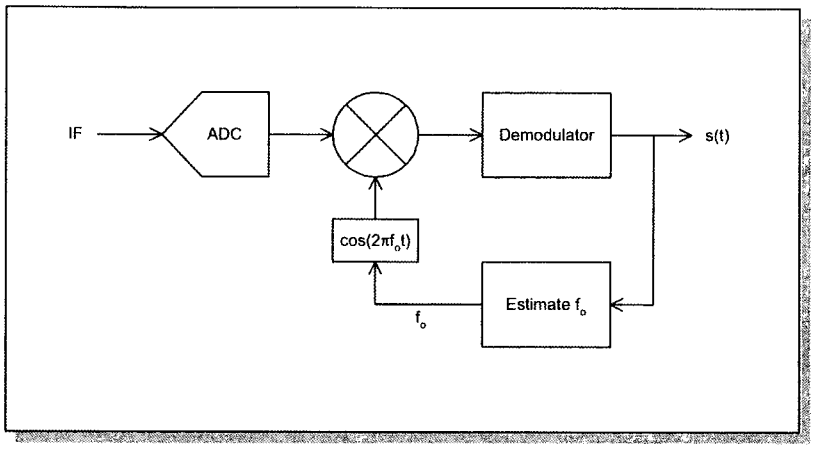


Figure 7.1: Frequency Correction IF Mixer

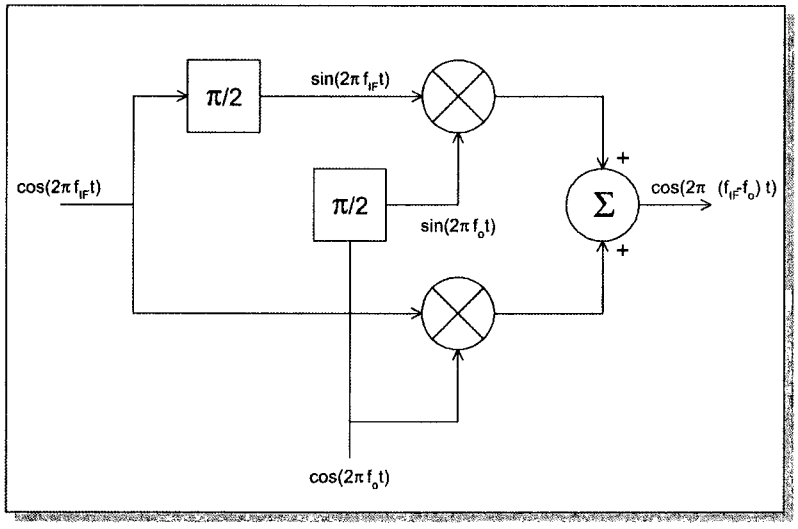


Figure 7.2: Balanced Modulator: Unwanted Mixer Product Cancellation

used in SSB-AM modulation, (see figure 7.2). This arrangement automatically cancels the unwanted upper sideband product from the mixers, whilst reinforcing the lower $(IF - f_o)$ sideband.

A critical requirement of a balanced modulator is the introduction of an exactly $\pi/2$ phase shift into the IF signal at all frequencies of interest, that is 500 kHz either side of the IF. Digitally this can be achieved with a Hilbert Transform bandpass IIR filter; giving the analytic equivalent of the IF input. Compensation for the group delay of the filter would also be needed in the non phase shifted branch. The $\exp(j2\pi f_o)$ components could be formed with a programmable phase accumulator and a sine coefficient look up table in the manner of a numerically controlled oscillator.

The second receiver stage, the demodulator, also offers some promising possibilities for frequency offset compensation, details of which depend upon the particular demodulation method being employed. However, they fall into two broad categories: either an additional operation to correct the signal as it is demodulated or a modification of the demodulation method itself. To illustrate with examples, the phase shift discriminator of figure 4.7 can be considered. Frequency offsets produce a linear ramp in the unwrapped phase signal $\phi(t)$ (see figure 6.1). Hong and Lee [66] propose removal of the frequency errors by finding the gradient of this ramp through regression and adding an appropriate linear term to correct the phase. Alternatively, as after differentiation the linear ramp reduces to a DC offset, an appropriate constant term could be estimated and subtracted from the demodulated signal. The same approach could be taken with the output of the quadrature detector (figure 4.6).

Adapting the demodulator process itself is generally not feasible in practice. For example, the structure of the quadrature detector's phase shift block could be modified to give a $\pi/2$ shift at $f_c - f_e$. If the discriminator is in the digital domain then the IF and sampling frequency will have been carefully chosen so that an integer number of sample delays will give the required phase shift at $f_c$ (i.e. at the

IF). Making corrections for values of $f_e$ in the kHz range is not possible[1]. In an analogue discriminator, correcting for $f_e$ would require that the RLC tank phase shift circuit be finely tuned to a slightly different value.

The final, and preferable, alternative is to accept the DC offset and provide a third receiver stage process, a decision algorithm, which is able to tolerate it without compromising BER. As described in section 4.2.2, there are a multiplicity of decision algorithms available to Bluetooth radio designers. At the heart of each, with the exception of a MLSE (see [46]), there is a comparator which attempts to resolve the current symbol's magnitude, as perceived by the demodulator, into a received digital bit. Alternatively, decisions algorithms can be viewed as bit slicers of varying complexity. For instance, DFE algorithms are bit slicers with a preprocessing stage that compensates for a degree of ISI. Therefore, for the remainder of the chapter, a simple bit slicer will be considered with the knowledge that techniques developed for it are directly applicable to the larger class of decision algorithms. Further justification for paying close attention to bit slicers is that they are utilised in the vast majority of commercial Bluetooth silicon.

The most natural way to compensate for frequency errors in a bit slicer is to dynamically adapt the decision threshold value, as suggested in [67] and perhaps implied by section 4.2.1 of the Bluetooth Specification [22]. This removes the influence of the DC level by adding its estimated magnitude to the default zero threshold. That is, the threshold value becomes the DC offset. The advantage of this approach is its simplicity. Beyond the calculation of the DC level, which is a necessary first step in estimating $f_o$, very little additional processing is required and, in contrast to the previous solutions, no operations are carried out on the demodulated signal. On balance, therefore, using an adaptive threshold bit slicer is the most attractive option. It is the simplest solution and will find applications in the largest number of designs. The following section will describe its implementation in detail and present

[1]This assumes that a reasonable sampling rate will be in the 20 MHz to 80MHz range.

simulations used to evaluate its performance and limitations.

## 7.2 Adaptive Threshold IaD Decision Algorithm

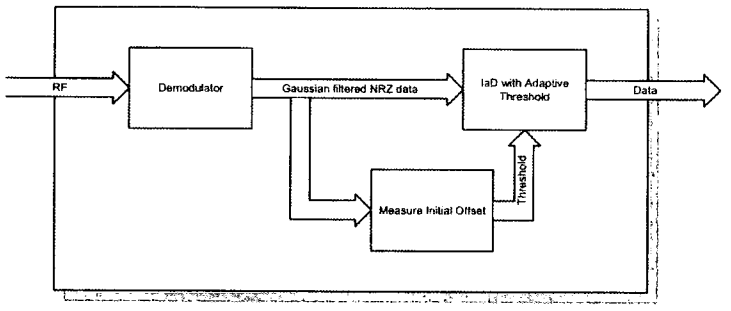


Figure 7.3: Adaptive Threshold GFSK Demodulator

The feed-forward structure of the adaptive threshold IaD is shown in figure 7.3. Demodulation proceeds in the normal way and the Gaussian filtered NRZ signal is passed to both the offset estimation process and the bit slicer. An initial offset is estimated from the first four received symbols, the packet preamble. During this estimation stage the bit slicer's decision threshold is still zero. No retrospective decisions need be made for the preamble, however, as they carry no useful information. The new threshold value is applied from the fifth symbol, the first of the packet header.

Clearly the bit slicer must be modified to accept a threshold value input and to enable selection of the appropriate decision threshold according to the symbol number in the current packet. A means of resetting the threshold value to zero between packets is also needed.

### 7.2.1 Adaptive Threshold Calculation

Calculation of the decision threshold is equivalent to the first stage of the initial frequency offset estimation method of section 6.2. However, here the desired result is the DC level of the preamble sample values. The most straightforward way of

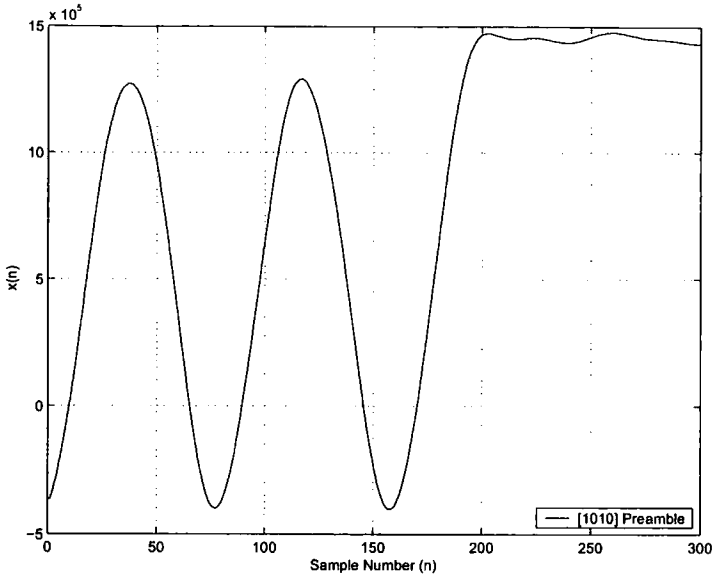


Figure 7.4: Sampled Packet Preamble, $f_o = -75$kHz, $f_s = 40$MHz.

calculating this is to assume that the waveform due to a "0" in the preamble is symmetrical to that due to a "1". Then, since there are always two of each bit in the preamble, the DC offset is simply equal to the numeric mean of the preamble's samples. An expression describing this process is:

$$Th = \frac{1}{N}\sum_{k=1}^{N} x(k), \tag{7.1}$$

where $Th$ is the new decision threshold, and N is the number of samples in the preamble $x(k)$. However, since decision algorithms do not necessarily utilise all samples in the demodulated data[2], a more general expression is:

$$Th = \frac{1}{N_d}\sum_{k=1}^{N_d} x_d(k), \tag{7.2}$$

where $N_d$ is defined in terms of a down sampling ratio $\alpha$:

$$N_d = 4\alpha\frac{f_s}{T}, \tag{7.3}$$

[2]Bit slicers of course use only 1 sample per symbol.

$$\frac{T}{f_s} \leq \alpha \leq 1. \tag{7.4}$$

$x_d(k)$ is obtained from $x(k)$ so that the number of samples per symbol in $x_d(k)$ becomes $\alpha(f_s/T)$. In the following sections, the two cases of $\alpha = 1$ and $\alpha = T/f_s$ will be considered. Thresholds will be calculated using all the preamble samples and only the central sample from each of the four symbols. The two cases represent the largest and smallest number of additions in the calculations respectively.

### 7.2.2 Performance Simulation

Investigations into the performance of the adaptive threshold IaD followed a similar simulation procedure to that described in chapter 5. The Matlab code was extended to include a threshold calculation, the result of which was utilised in the revised decision algorithm for the fifth and subsequent bits of a packet (see appendix A.3). Again DH5 packets were generated with uniformly random distributed data and modulated onto a nominally 2 MHz carrier frequency at a sample rate of 320 MHz. Initial frequency offset errors were introduced into the modulated signal as before, with values of 0 kHz to 80 kHz in 10 kHz steps and additionally 150 kHz. AWGN noise was added to give $E_b/N_o$ from 11 dB to 22 dB in 0.5 dB steps. This range covers the BERs of interest, approximately two orders of magnitude above and below $10^{-3}$. Again the simulation was run for 100 cycles at each combination of $E_b/N_o$ and $f_o$.

For consistency with chapter 5, both the phase shift discriminator (PSD) and quadrature detector (QD) demodulators were simulated. Therefore, there are four sets of results: PSD and all samples threshold; PSD and central samples threshold; QD and all samples threshold; and finally the QD with a central samples only generated threshold. In this way an objective assessment can be made of the threshold calculation methods' relative performance.

A minor modification was made to the BER calculation routine. In the previous

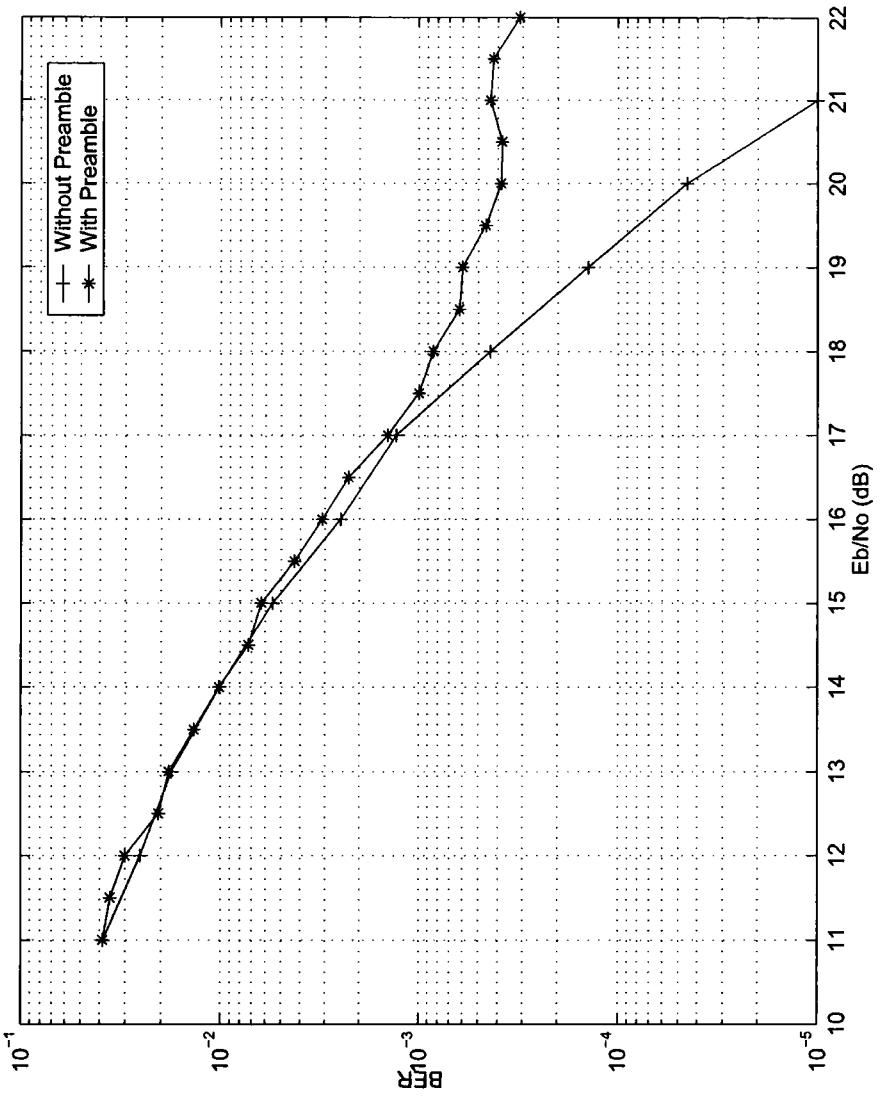


Figure 7.5: Effect of Errors in the Preamble on BER, $f_o = 80$ kHz.

| Demodulator | Calculation Method | Minimum $E_b/N_o$ (dB) | Improvement (dB) |
|---|---|---|---|
| QD | Central Sample | 16.1 | 13.9 |
| QD | All Samples | 16.4 | 13.6 |
| PSD | Central Sample | 17.0 | 13.0 |
| PSD | All Samples | 16.6 | 13.4 |

Table 7.1: Performance Improvements at $f_o = 70$kHz.

simulations errors in all of the received symbols were included. However, in these tests the four preamble bits were detected differently to the rest of the packet. The effect of this was to artificially inflate BERs at high SNR and high $f_o$. Indeed, initial results with $f_o = 150$ kHz had the BER asymptotic to approximately $1.5 \times 10^{-3}$ with increasing SNR. This was because the only bits in error were then in the preamble. A similar but less severe effect is shown in figure 7.5 for $f_o = 80$ kHz. Consequently, all BERs have been calculated by excluding errors in the preamble. This can be done without compromising the interpretation of the results, since the excluded bits do not bear information used by higher protocol layers.

## 7.3 Simulation Results

The adaptive threshold IaD simulation results are presented in figures 7.10 to 7.14. By referring to these figures it is clear that the technique successfully compensates for initial carrier frequency offset errors. The performance degradation due to $f_o =$ 150 kHz is not greater than 2.7 dB. This is significantly less than the degradation of 14.25 dB previously observed with no frequency compensation and $f_o = 70$ kHz. Table 7.1 summarises the results for $f_o = 70$ kHz and the resultant improvement in performance for each of the four combinations of demodulation algorithm and threshold estimator.

Table 7.3 gives results in terms of the minimum $E_b/N_o$ required for a BER of $10^{-3}$ at various values of $f_o$. The data in this table pertaining to the QD displays a curious trend. At low $f_o$ the central sample threshold calculation outperforms the all

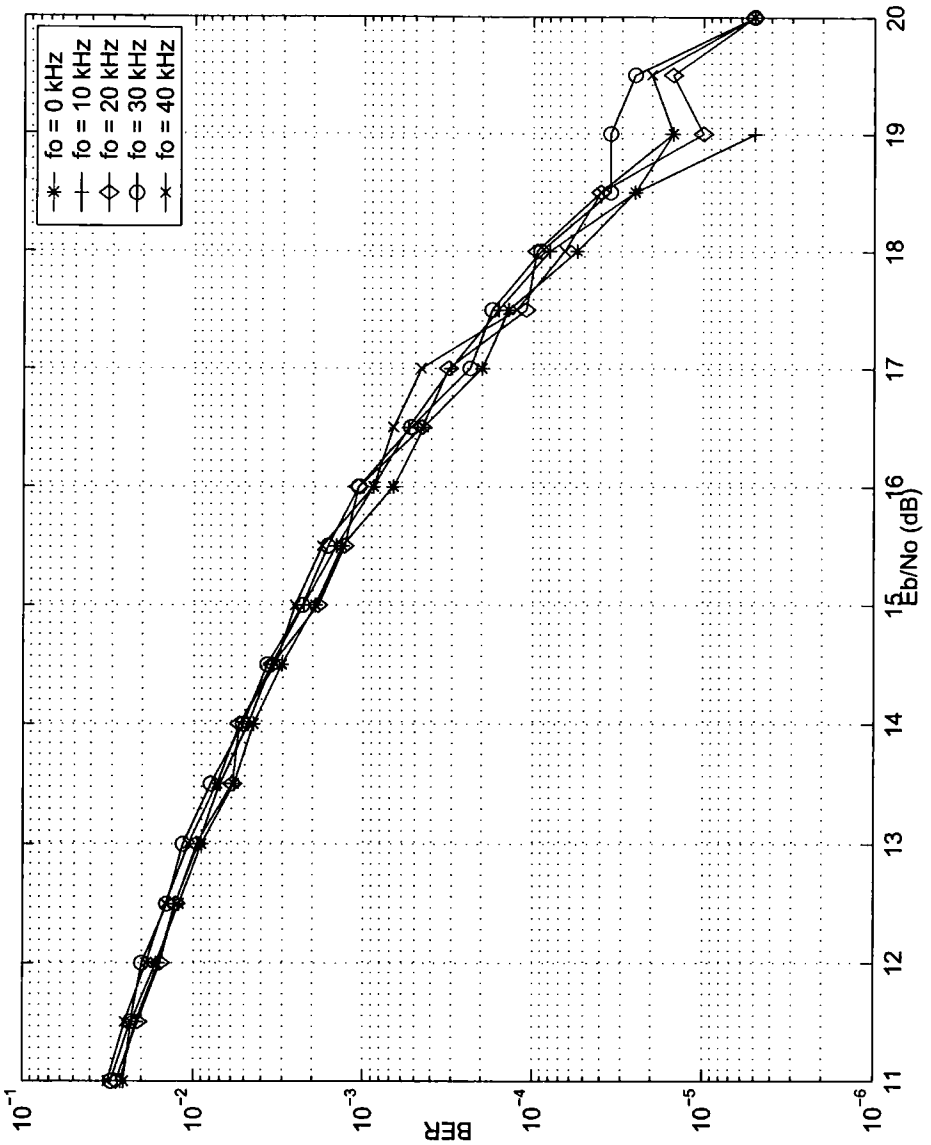


Figure 7.6: QD, All Samples, $f_o = 0$ to 40 kHz.

| Demodulator | Calculation Method | $E_b/N_o$ (dB) | | |
|---|---|---|---|---|
| | | $f_o = 0$kHz | $f_o = 80$kHz | $f_o = 150$kHz |
| QD | Central Sample | 15.4 | 16.3 | 18.1 |
| QD | All Samples | 15.7 | 16.5 | 18.0 |
| PSD | Central Sample | 16.5 | 17.0 | 17.2 |
| PSD | All Samples | 16.1 | 16.6 | 17.0 |

Table 7.2: Minimum $E_b/N_o$ for BER of $10^{-3}$ at various $f_o$.

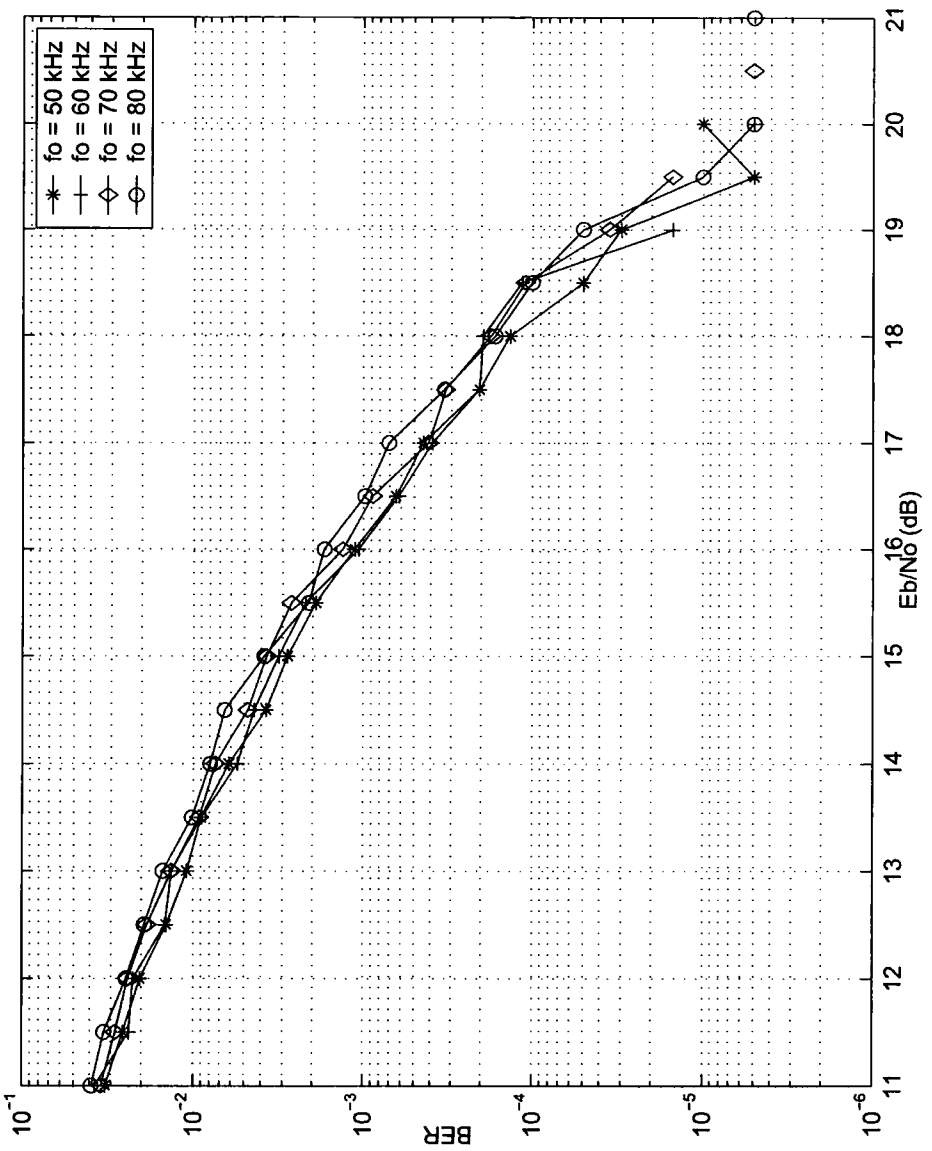


Figure 7.7: QD, All Samples, $f_o = 50$ to 80 kHz.

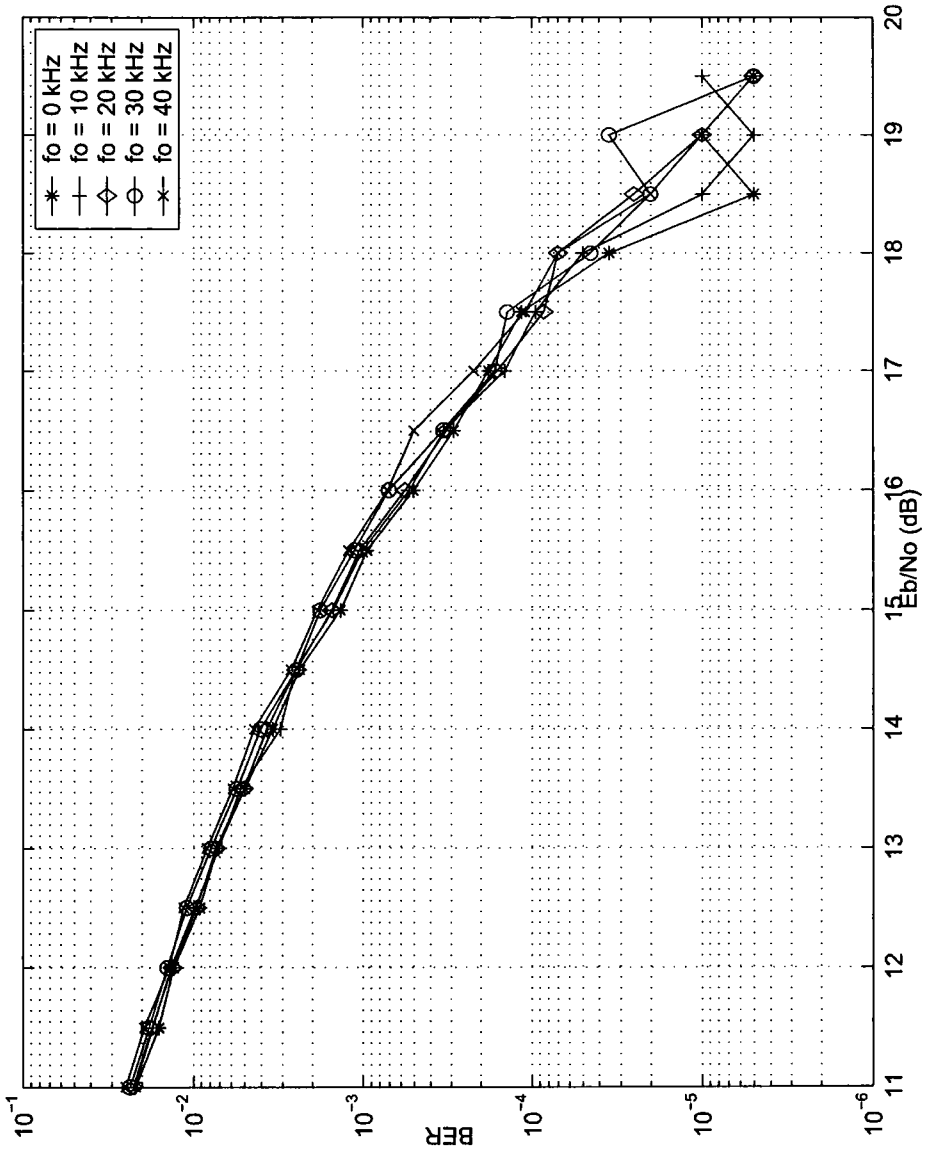


Figure 7.8: QD, Central Samples, $f_o = 0$ to 40 kHz.

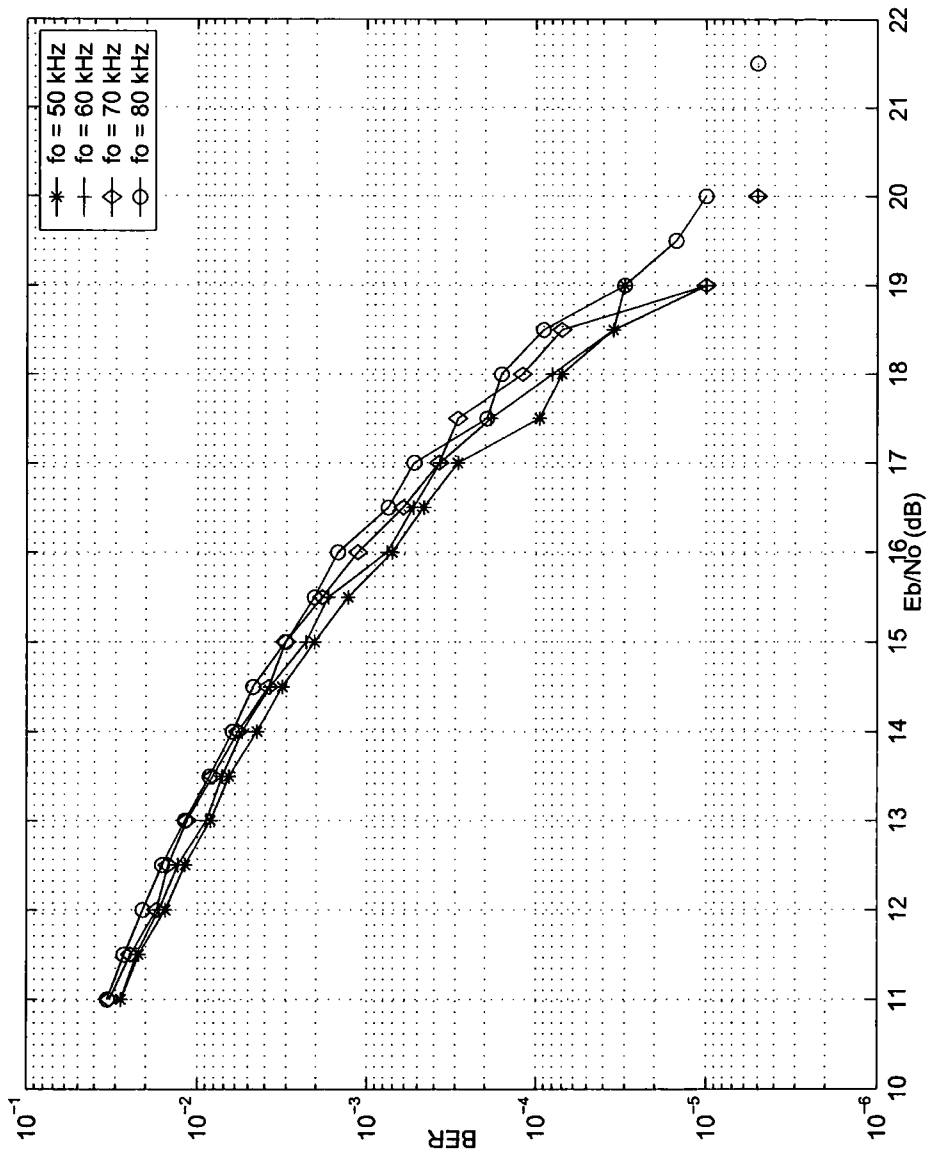


Figure 7.9: QD, Central Samples, $f_o = 50$ to 80 kHz.

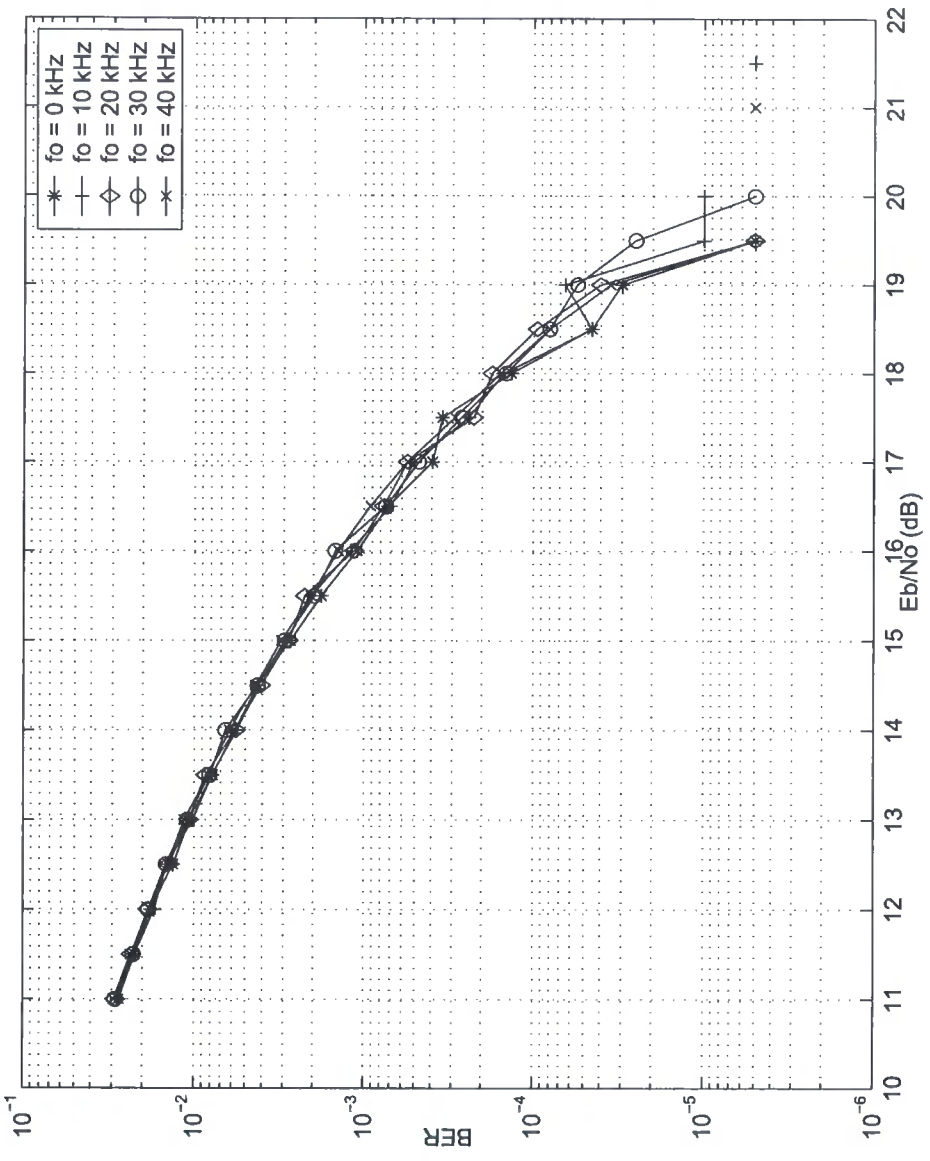


Figure 7.10: PSD, All Samples, $f_o = 0$ to 40 kHz.

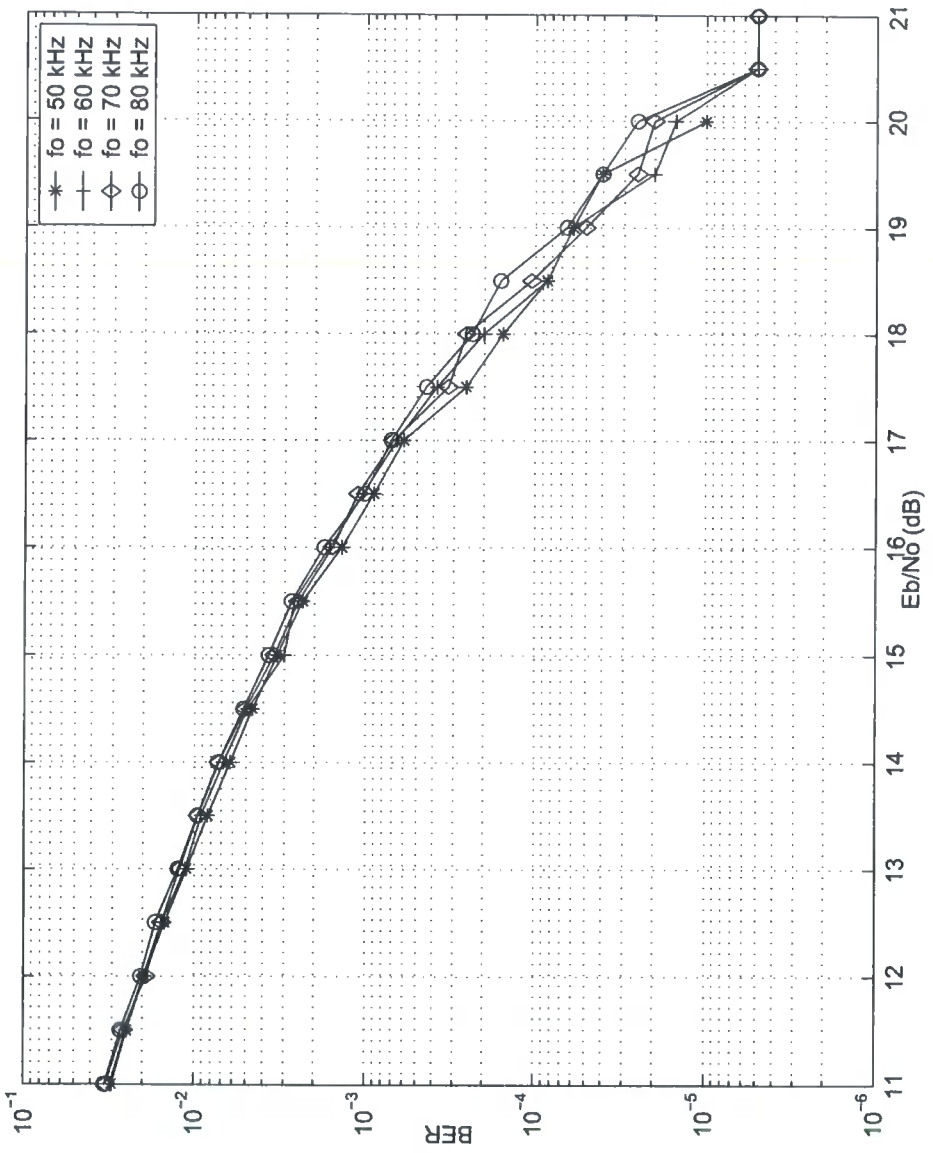


Figure 7.11: PSD, All Samples, $f_o = 50$ to 80 kHz.

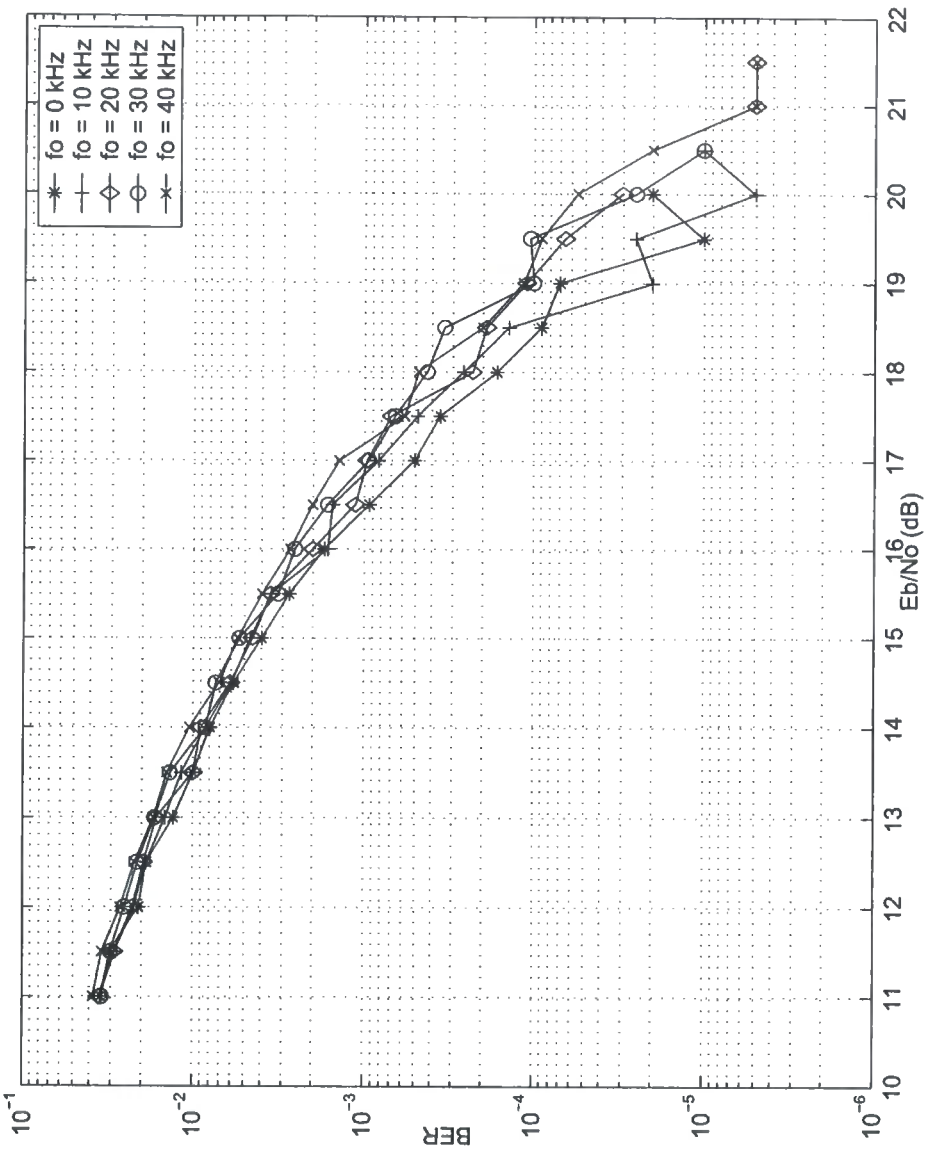


Figure 7.12: PSD, Central Samples, $f_o = 0$ to 40 kHz.

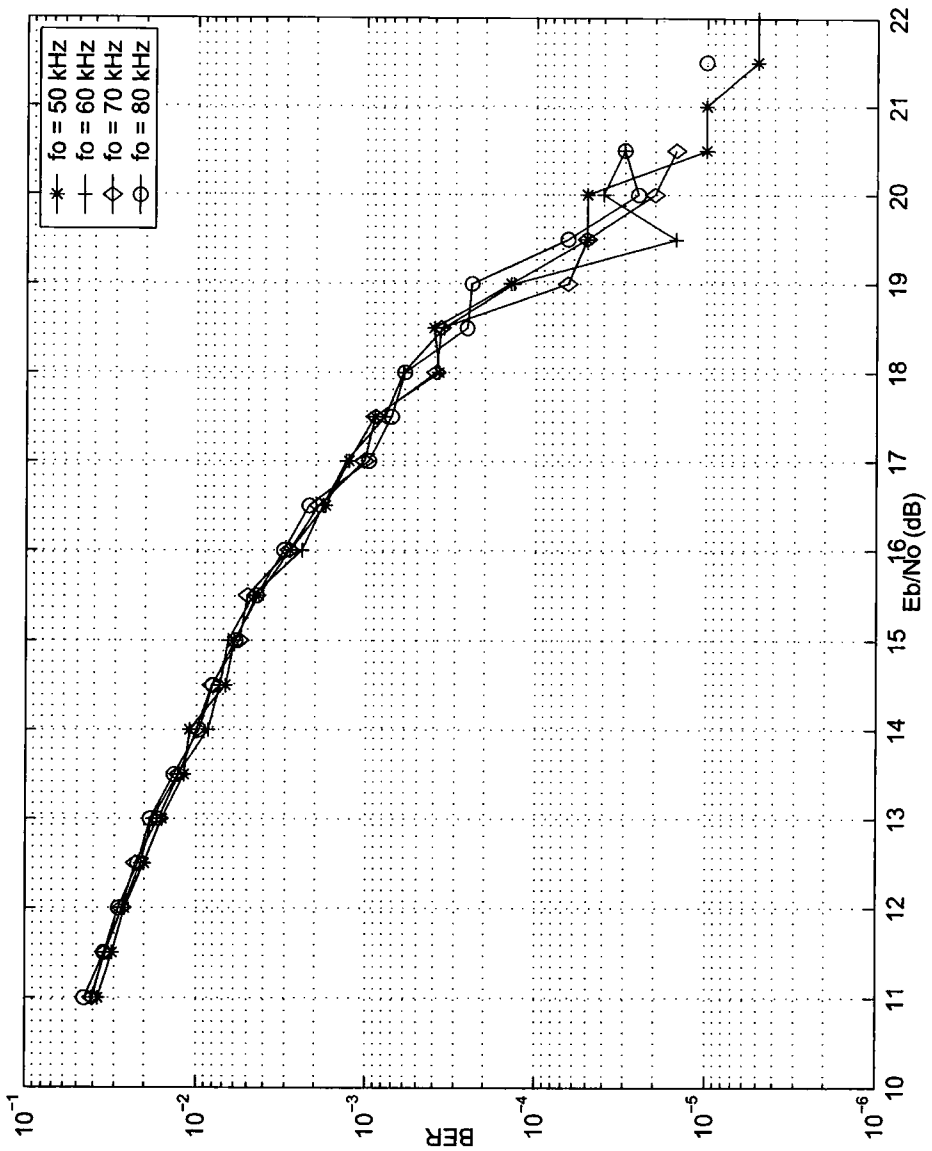


Figure 7.13: PSD, Central Samples, $f_o = 50$ to 80 kHz.

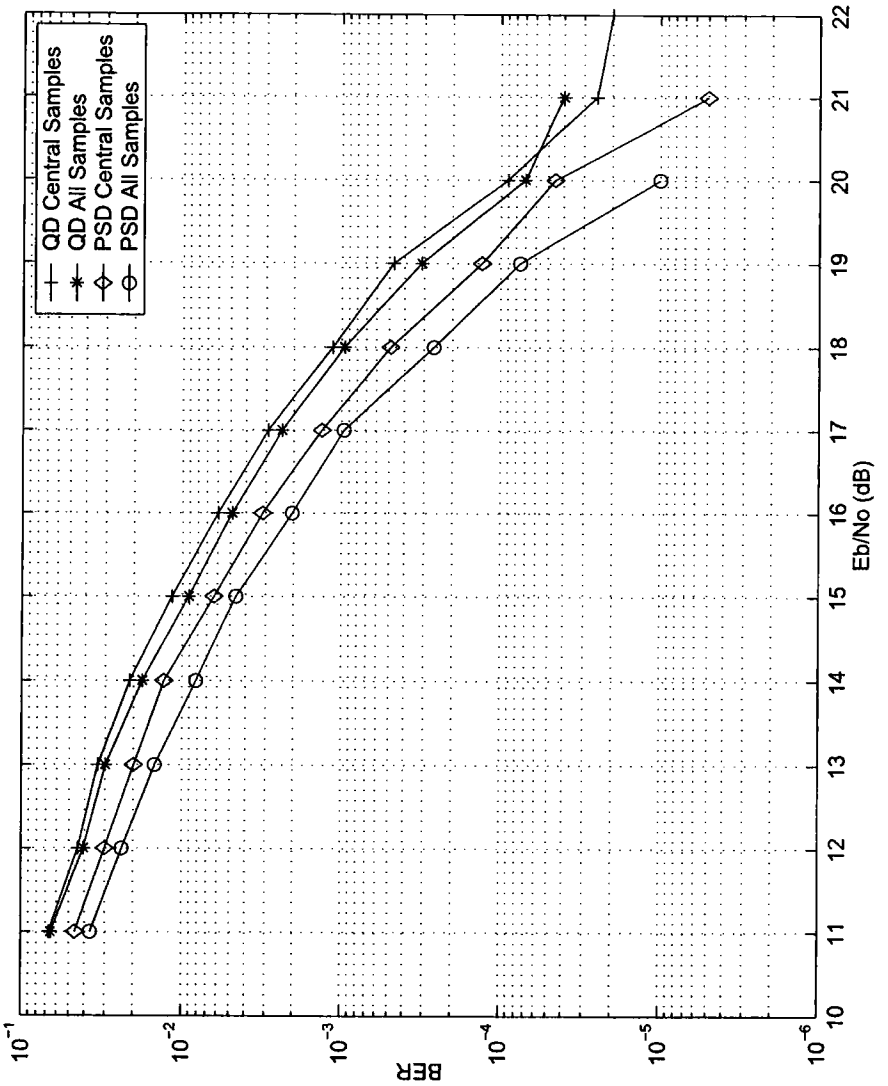


Figure 7.14: BER with $f_o = 150$ kHz.

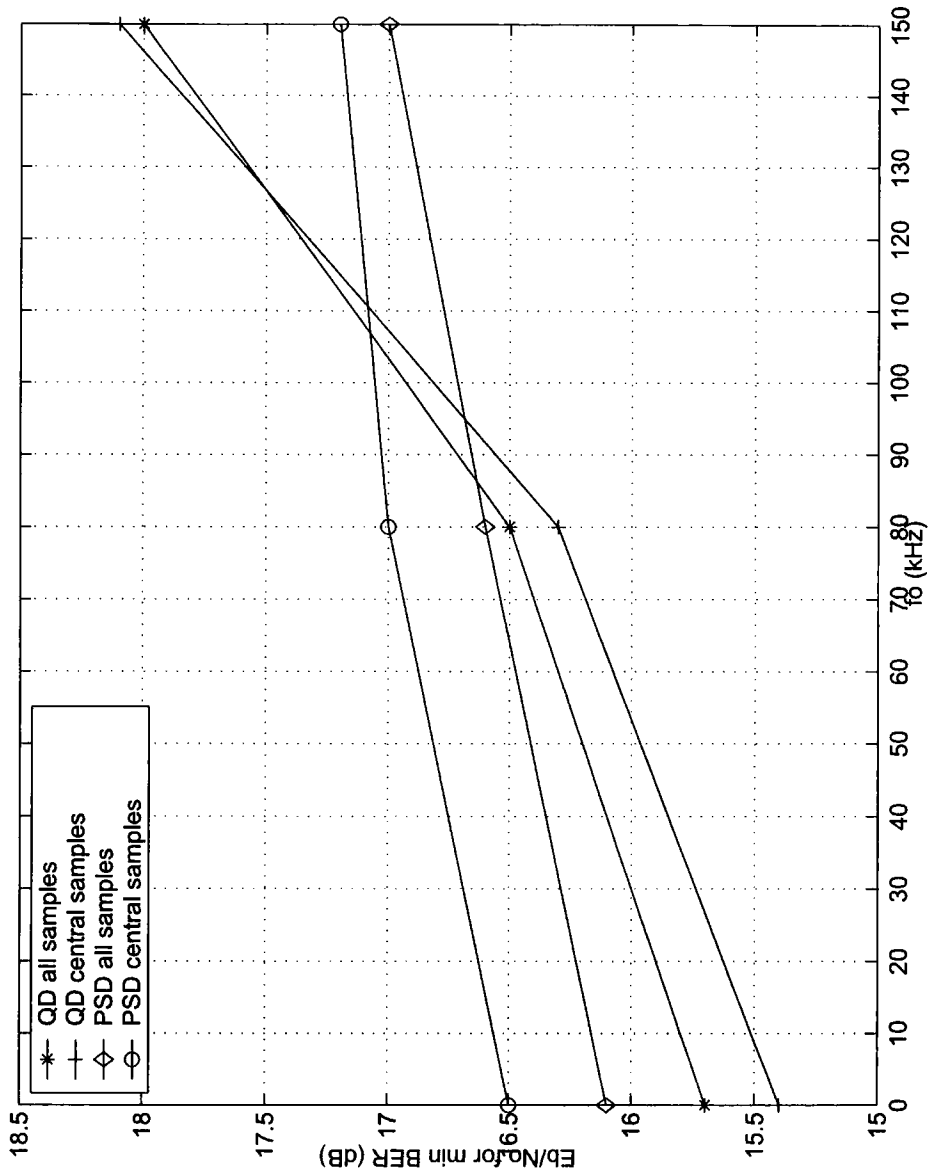


Figure 7.15: $E_b/N_o$ Required for $10^{-3}$ BER versus $f_o$.

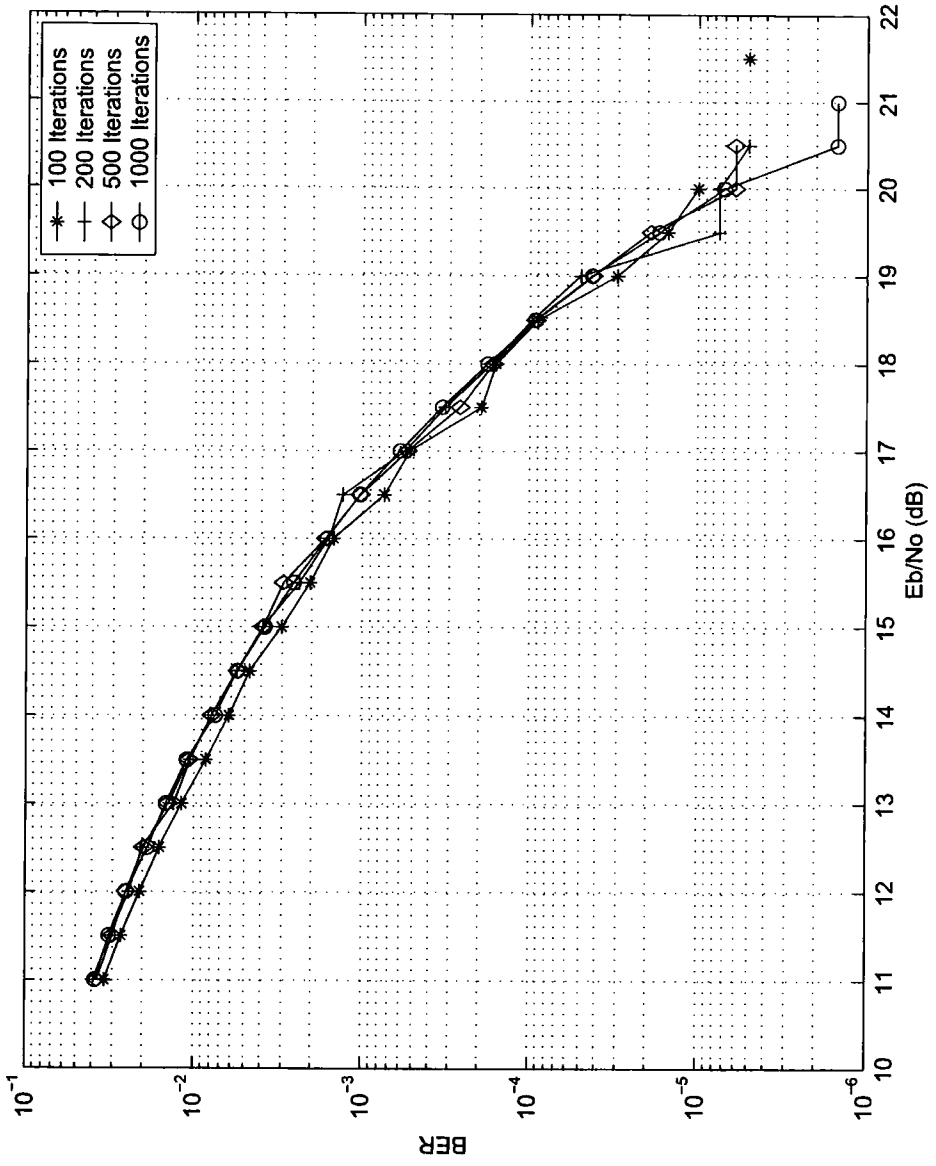


Figure 7.16: BER Results for increasing numbers of simulator iterations per $E_b/N_o$ value. QD demodulator, Central Samples, $f_0 = 80$ kHz.

samples calculation. This is the opposite of what would intuitively be expected and indeed is opposite to the behaviour displayed in the PSD results. Figure 7.15 shows this more clearly. Two possible explanations for this are errors in the calculation of the BERs and/or the dependance of the QD output on input signal amplitude. In order to evaluate the effect of errors in the BER values, the QD (and PSD) with central sample threshold simulations were rerun with $f_o$ = 80 kHz and 200, 500 and 1000 iterations per SNR data point. This gave the new plots shown in figure 7.16 (and 7.17). The plot for 1000 iterations is smoother, being averaged over a greater number of trials, and gives a minimum $E_b/N_o$ value of 0.23 dB greater than the plot for 100 iterations. Similar results for the QD and all samples threshold calculation show an increase in minimum $E_b/N_o$ of 0.1 dB. This means that the true performance differential between the threshold calculation methods is closer to 0.07 dB rather than the 0.2 dB reported in table 7.3. It is reasonable to conclude, therefore, that the performance of the two methods with the QD is essentially the same, and that the trend described above is due to errors in the BER estimates.

Figure 7.15 also demonstrates that, overall, the QD is better than the PSD at low $f_o$ and with adaptive thresholding. This situation reverses between approximately 85 kHz and 110 kHz.

Whilst the adaptive threshold IaD offers significant BER improvements at large $f_o$, there is an approximately 1 dB degradation in performance for $f_o < 10$kHz. This is due to the errors in the calculation of the threshold value at low SNR. Both methods assume that the demodulated message is symmetrical in the preamble, which was not the case as noise was present throughout the packet, including in the preamble.

## 7.4 Conclusions

The adaptive decision threshold bit slicing decision algorithm has been shown to be both effective and simple. The full range of frequency offsets which may be encountered by a compliant device are compensated for, with a maximum performance degradation of 2.5 dB with a 150 kHz error.

Calculating the new, adapted decision threshold is optimally achieved by taking only the central sample of each preamble symbol into consideration. With a QD demodulator there is no practical difference between this method and the more complex alternatives ($\alpha < T/f_s$). With a PSD there is at most a 0.4 dB penalty. However, this is balanced by a simpler implementation consisting of an accumulator with a lower update rate and a divide by four operation. Of course, in the digital domain division by four is equivalent to a right shift by two bit positions. That is, the two LSBs of the accumulator output are discarded.

The degradation in performance due to the adaptive IaD with $f_o < 10$ kHz is more than outweighed by the improvement at higher $f_o$. Also, if the demodulator gives an output independent of the magnitude of its input, as with the PSD but not the QD, then the degradation could be removed by using a zero threshold when the estimated DC offset is less than that which would be due to $f_o = 10$ kHz.

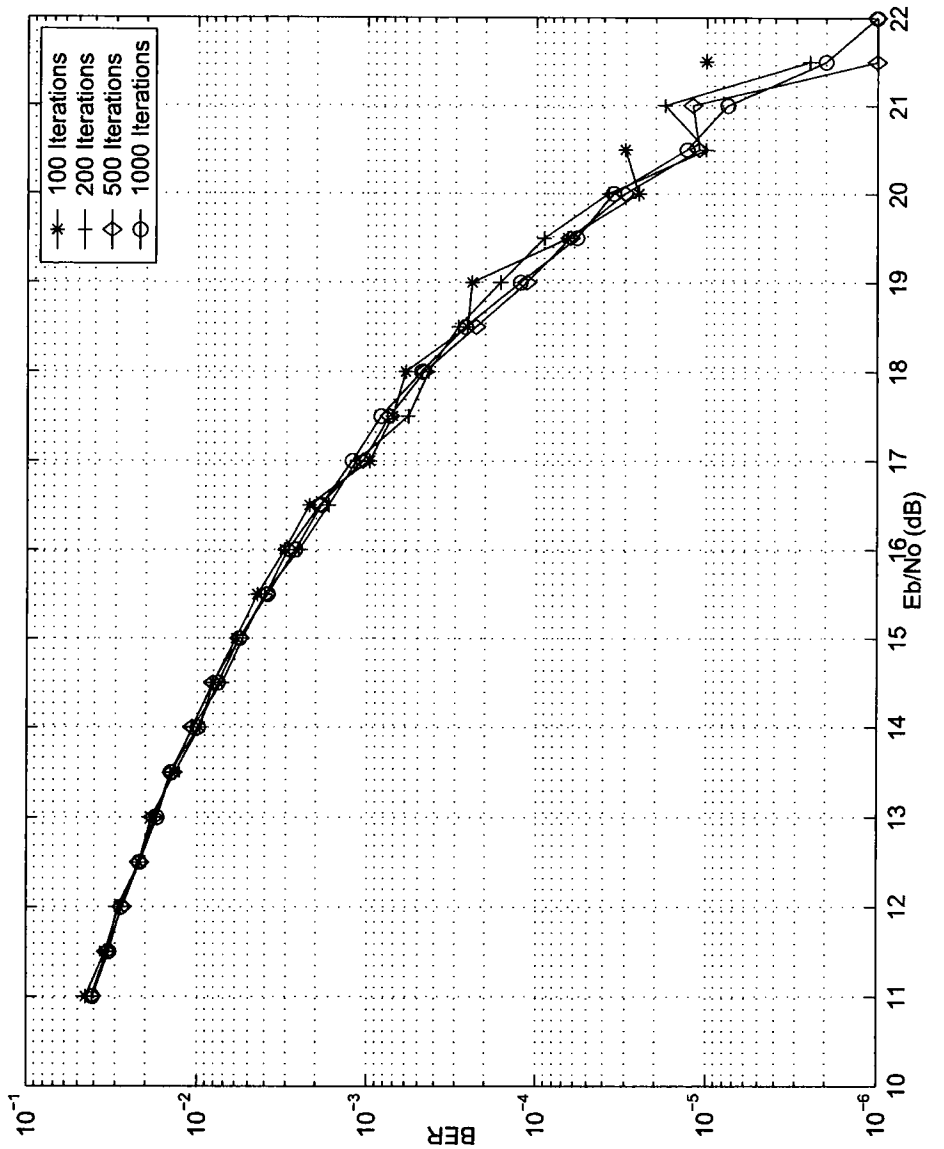


Figure 7.17: BER Results for increasing numbers of simulator iterations per $E_b/N_o$ value. PSD demodulator, Central Samples, $f_0 = 80$ kHz.

# Chapter 8

# In-house Bluetooth Tester

## 8.1 Motivation and Specification

Chapter 5 described the effect of carrier frequency errors upon Bluetooth link performance, whilst chapter 6 introduced a method for the measurement of such errors. This chapter will describe the design of a VXIBus instrument, developed in the School of Engineering. It is intended to be able to verify the results of the simulations reported in chapter 5. This work was also prompted by Integrated Measurement Solutions (IMS, now part of Credence Inc) who expressed an interest in a Bluetooth test capability for their mixed signal products.

### 8.1.1 Existing Mixed Signal Test Facilities

IMS' early involvement in the project informed the choice of the School of Engineering's Mixed Signal Test Station (MSTS) as a development platform. The MSTS is an integrated circuit engineering test validation system. Validations systems are distinct to the automatic test equipment used in production environments. In ATE the emphasis is upon test time and reducing the cost of test. This leads to highly optimised systems that are designed to provide pass/fail decisions on DUTs and are difficult to programme. In contrast validation systems do not have to be very fast.

Instead desirable properties are detailed test results, ease of use and an intuitive interface.

The MSTS is partitioned into a digital and an analogue section. The digital half stimulates a DUT's logic interface and receives data from it. The analogue instrumentation is housed in a VXIBus[1] mainframe and provides analog waveform generation and capture capabilities. There are three sets of instruments; an audio frequency generator, analyser, switching matrix and filter bank; a video frequency set; and an RF set which can provide signals at up to 2.5 GHz and capture at 500 Ms/sec. All three instrument sets have clocks and triggers derived from a VXIBus sychronisation module which in turn takes its reference from a highly stable 1 GHz clock. The digital sections clocks are also derived from the synchronisation module. This allows coherent capture when testing components such as ADCs. The RF instruments, synchronisation module and DUT interface boards are new and bring the MSTS up to the standard of IMS's latest mixed signal system called the Electra.

A key strength of the MSTS is the ease with which a test programme can be written and changed, allowing engineers to ask 'what if' questions during silicon debug. This flexibility is facilitated through the design of the interface software which is written with National Instruments' Labview.

### 8.1.2 Functional Description

The Bluetooth test module must fulfill three basic requirements:

- Be able to act as the master device in a Bluetooth Link Manager test.
- Provide control over the output RF frequency independently of any Bluetooth protocol considerations.
- Provide access to signals from a DUT at a frequency suitable for capture with

[1]V(ME) eXtensions for Instrumentation Bus. VME is a computer backplane standard.

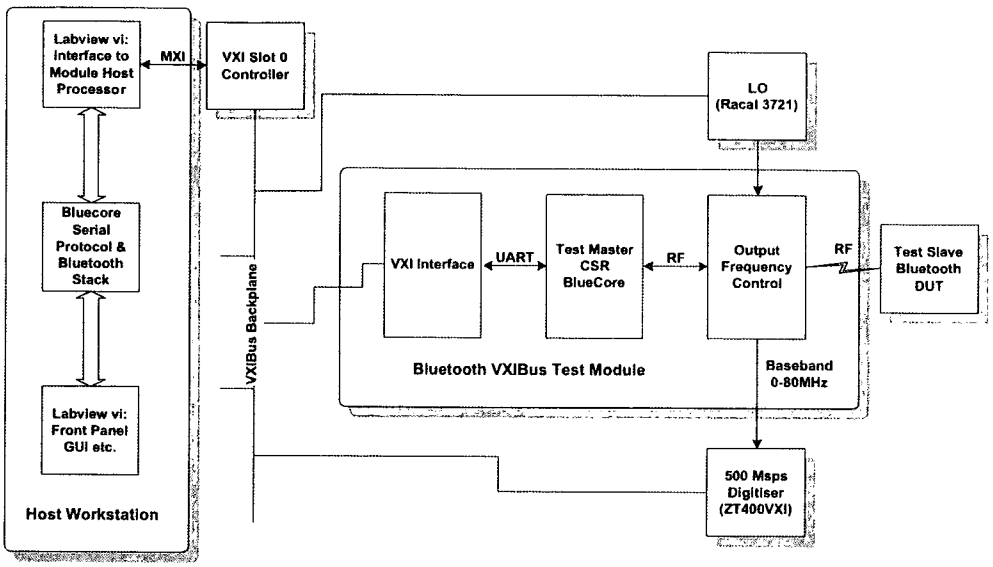


Figure 8.1: Bluetooth Tester: High level functional blocks.

an ADC so that they may be analysed for carrier frequency tolerances.

The design of the Bluetooth tester is based upon a VXIBus module integrated into the MSTS' existing analogue instrumentation and VXI mainframe. The advantages of this approach are ease of interaction with MSTS' standard instrument package through the VXI backplane, communications with a host workstation via the VXI controller and the possibility of developing the Bluetooth tester's user interface in the Labview programming language.

IMS provide Labview graphical interfaces for all of its analogue and digital hardware. This enables complex tests to be constructed by connecting appropriate sub routines, called virtual instruments or VIs, in a natural way using Labview's "G" graphical programming method. Typically a test of a mixed signal device will require the cooperation of several instruments, for instance digital pin electronics, an ADC, clock timing and triggering electronics to test a DAC part. Using Labview all of these elements can be setup within a single controlling VI. This requires, however, that the instruments used have appropriate control VIs. Therefore, in order to be integrated into the MSTS any new design has to have its own VIs to control

all aspects of the instrument from within a test programme. For this reason the Bluetooth tester's user interface must be in Labview. This has some distinct advantages, not least Labview's built in routines for communication with VXI devices, but also presents a problem; the upper layers of the Bluetooth protocol stack that are needed to form a complete Bluetooth device must also interface with Labview. These software portions of the design are discussed further in section 8.2.4.

The Bluetooth tester is split into five functional blocks:

- The master Bluetooth device and its interface with the VXI backplane and host workstation.
- The electronics required to control the frequency at the DUT's RF interface and to provide access to these signals for capture by an ADC.
- The software interface to the master Bluetooth device and a user.
- Control functions.
- External instruments.

Figure 8.1 illustrates the relationship between the functional blocks. As can be seen the frequency shifting electronics enables access to the DUT's air interface at baseband frequencies.

The next section describes these five functional elements in detail.

## 8.2 Design Details

### 8.2.1 Bluetooth Test Master

A prerequisite for performing radio tests on a Bluetooth DUT, as described in section 3.5, is a further Bluetooth device to act as the master in a piconet. In our test system this master device is a BlueCore module from Cambridge Silicon Radio. There is nothing particularly special about the BlueCore radio, it was chosen for its

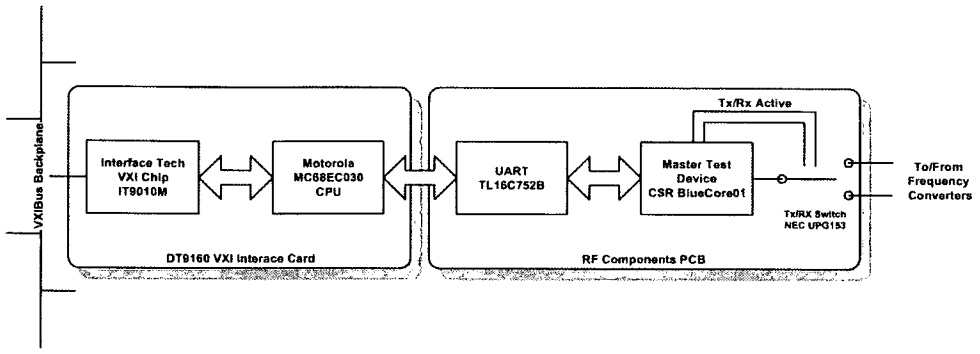


Figure 8.2: Test Master Connections

availability through academic contacts with CSR and proven reliability. However, the need to have a Bluetooth radio in the VXI module presents the first design challenge: how to connect the chip to the outside world?

As explained in chapter 3 most Bluetooth devices need to communicate with a host computer upon which is running the upper layers of the Bluetooth protocol stack. Indeed the protocol stack has a special layer, namely the Host Controller Interface, which facilitates this requirement. Only very simple devices, such as mobile phone headsets, with correspondingly simple application layers need not have a host, a radio intended as a master test device does require one [68]. The connection between the BlueCore and its host must therefore, in our case, be at the HCI layer, and furthermore, be transparent to both sides. Normal HCI communication is across a UART or USB link. However, no direct serial connection is possible; the HCI layer must span the VXI-MXI bridge.

Figure 8.2 illustrates the hardware side of the solution. The BlueCore has an internal UART interface so that it may be built into peripherals such as serial port dongles. This is connected, in the normal way, to a serial UART IC, except that the connection is upon a PCB rather than across a serial cable. The microprocessor interface of the UART IC is then wired, with the addition of a small amount of glue logic, into the address space of the Motorola 68030 microprocessor of the VXI interface card. In a normal embedded product, such as a PDA, that might be

sufficient as the upper layers of the Bluetooth protocol would reside upon the 68030. However, here the code on the microprocessor is fixed and is concerned only with interpreting messages to and from the VXI backplane. Two such messages are VXI reads and writes into the address space of the 68030, so by using the correct addresses direct access to the UART's internal registers and FIFOs is possible. The hardware of the DT9160 has address decoding logic to ease this task. Special user address region select signals are provided to indicate to external chips, such as the UART, that an access is in progress. Using the user address lines as a chip select and the three least significant address bits of the 68030 is sufficient to index the UART's registers. So, at the hardware level, data is written to and read from the BlueCore by writing and reading to the appropriate UART FIFOs by accessing the correct addresses in the address space of the VXI module.

The layers of the Bluetooth protocol stack at the HCI and below execute upon an embedded processor in the BlueCore IC which takes its instruction data from a Flash ROM chip on the BlueCore module PCB. The Flash memory is programmed during manufacture of the module. Section 8.2.4 goes on to describe the software on the host computer, a Sun Ultra 10 / Solaris 7 workstation. At the lowest layer this code consists of Labview VIs passing data to and from the UART, and at the highest layer a VI for the user's interface to the Bluetooth software stack.

### 8.2.2 Frequency Conversion

Control of the tester output frequency is achieved with two pairs of direct down-conversion and up-conversion mixers arranged as shown in figure 8.3, and controlling their local oscillator (LO) frequencies to give the desired result. If $S_{IN}$ is the input signal of frequency $f_{IN}$, $S_{OUT}$ the output signal with frequency $f_{OUT}$, $f_{LO1}$ the down-conversion LO and $f_{LO2}$ the up-conversion LO, then this process may be expressed

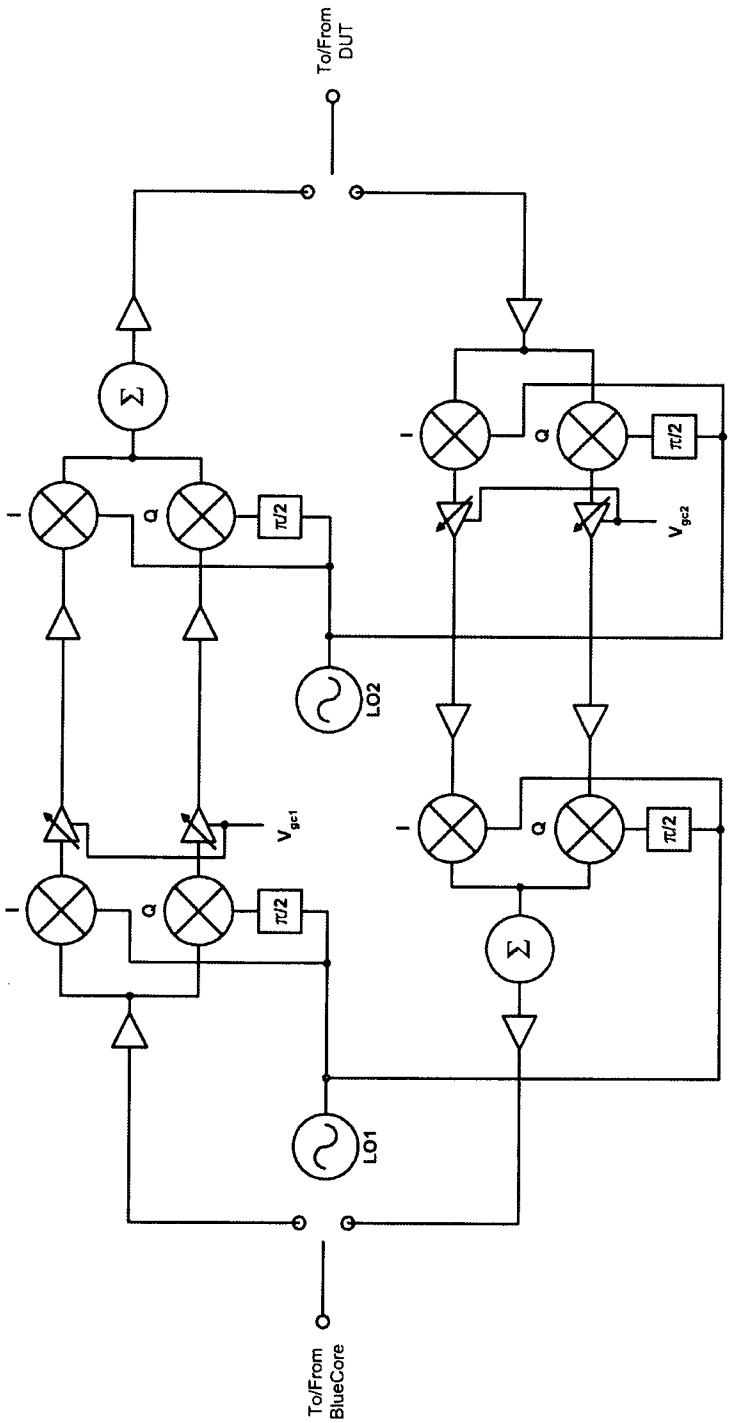


Figure 8.3: Transmit and Receive Frequency Converters

as:

$$S_{IN}(t) = \cos(2\pi f_{IN} t)$$

$$\begin{aligned} S_{BB}(t) &= \cos(2\pi f_{IN} t)\cos(2\pi f_{LO1} t) \\ &= \frac{1}{2}\cos(2\pi(f_{IN} - f_{LO1})t) + \frac{1}{2}\cos(2\pi(f_{IN} + f_{LO1})t) \\ &= \frac{1}{2}\cos(2\pi(f_{IN} - f_{LO1})t) \text{ after low pass filtering,} \end{aligned}$$

$$\begin{aligned} S_{OUT}(t) &= \cos(2\pi(f_{IN} - f_{LO1})t)\cos(2\pi f_{LO2} t) \\ &= \frac{1}{2}\cos(2\pi(f_{IN} - f_{LO1} + f_{LO2})t) + \frac{1}{2}\cos(2\pi(f_{IN} - f_{LO1} - f_{LO2})t) \\ &= \frac{1}{2}\cos(2\pi(f_{IN} + (f_{LO2} - f_{LO1}))t) \text{ after low pass filtering,} \qquad (8.1) \\ &= \frac{1}{2}\cos(2\pi f_{OUT} t). \end{aligned}$$

Therefore,

$$f_{OUT} = f_{IN} + (f_{LO2} - f_{LO1}). \qquad (8.2)$$

For instance if $f_{IN} = 2.405$ GHz, $f_{LO1} = 2.4$ GHz and $f_{LO2} = 2.4005$ GHz then

$$f_{OUT} = 2.405 \times 10^9 + (2.4005 \times 10^9 - 2.4 \times 10^9) = 2.4055 \text{ GHz},$$

that is an increase in frequency of 500 kHz with respect to $f_{IN}$.

Two pairs of mixers have been used in order to alter the frequency of signals both to and from a DUT. For convenience signals from the BlueCore to the DUT are said to be in the *transmit* direction, whilst signals from the DUT to the BlueCore are travelling in the *receive* direction. Control of frequencies in both directions is provided to mitigate against possible DUT carrier frequency errors. The ability to correct for large carrier offsets, that is greater than those allowed in the Bluetooth specification (section 3.3.1 and [22]), is retained as the BlueCore will receive data

in the correct frequency range. Also the mixers pairs will alter the amplitude of the signals over that which might be expected from free space or direct cable connections; using them in both directions balances this difference. The mixers used are Maxim MAX2701 direct down-conversion receivers and MAX2721 direct up-conversion receivers, which were procured pre-assembled onto PCBs in evaluation kit form.

In addition to their frequency conversion components both of the Maxim ICs have internal amplifiers. In the case of the MAX2701 receiver there is a low noise amplifier on the front end and a matched pair of variable gain amplifiers for the complex baseband signals. In this design the VGAs are used to independently control the RF signal levels at the DUT and BlueCore, and also to compensate for the attenuation which inevitably results from the addition of multiple components in the signal path. The level of control over the output amplitude afforded by the VGAs was measured with a VNA and found to be approximately 30 dB , between -35 dB and -5 dB . There is still an overall attenuation at the up-converter output but it is an acceptable level.

The primary reason for providing the ability to alter frequencies in the transmit direction is to evaluate the performance of a DUT under carrier frequency offsets and drifts. By keeping $f_{LO1}$ constant and altering $f_{LO2}$ using the Racal 3721 RF source in the VXI mainframe[2], arbitrary carrier errors can be introduced. $f_{LO1}$ is generated with a Wiltron 6817B bench top source, set constantly to output a signal at 2.4 GHz which was checked with an external oscilloscope (Tektronix TDS7254).

From figure 8.3 it is clear that the RF signals must be switched between the mixer pairs during transmit and receive cycles as appropriate. This is achieved with UPG153 GaAs solid state switches from NEC [69], controlled by the BlueCore's transmit and receive active lines [70], which are also buffered, with Burr Brown 634 unity gain amplifiers, to two external trigger lines on the VXI backplane (see section 8.2.5).

[2]RF power splitters are used to distribute a LO from a single source to two mixers simultaneously.

#### 8.2.2.1 Base-Band Access

A useful side effect of the *down-conversion – up-conversion* process is access to signals at baseband frequencies between the mixers. Figure 8.3 shows the complex outputs of the down-converters connected directly to the inputs of the up-converters. This is not actually the case as figure 8.4 shows. The in-phase (I) output of the down-converter is connected to a pair of high frequency relays. One, relay 2, connects to the I input of the up-converter whilst the other, relay 1, connects to a third relay, relay 9. This relay also has an input from the I output of the receive direction down-converter and its output is connected to an external 50 Ω SMA type jack. This connecter is the base-band output of the tester. For the quadrature signal two relays are also used, one to switch into the up-converter and another which simply switches into a 50 Ω load, with the Q relays following the state of their I counterparts. This arrangement is to ensure that whatever the connections for I are, Q will always see the same load, thus preventing amplitude mismatches between I and Q at the up-converter. Relays 5,6,7, and 8 perform the same functions in the receive direction. With this switching arrangement a combination of the following configurations is possible (see also Table 8.2.3 for a summary):

- Transmit direction:
  - BlueCore connected to up-converter (and therefore the DUT) only.
  - BlueCore connected to base-band output only.
  - BlueCore connected to up-converter and base-band output.
- Receive Direction:
  - DUT connected to up-converter (and therefore the BlueCore) only.
  - DUT connected to base-band output only.
  - DUT connected to up-converter and base-band output.

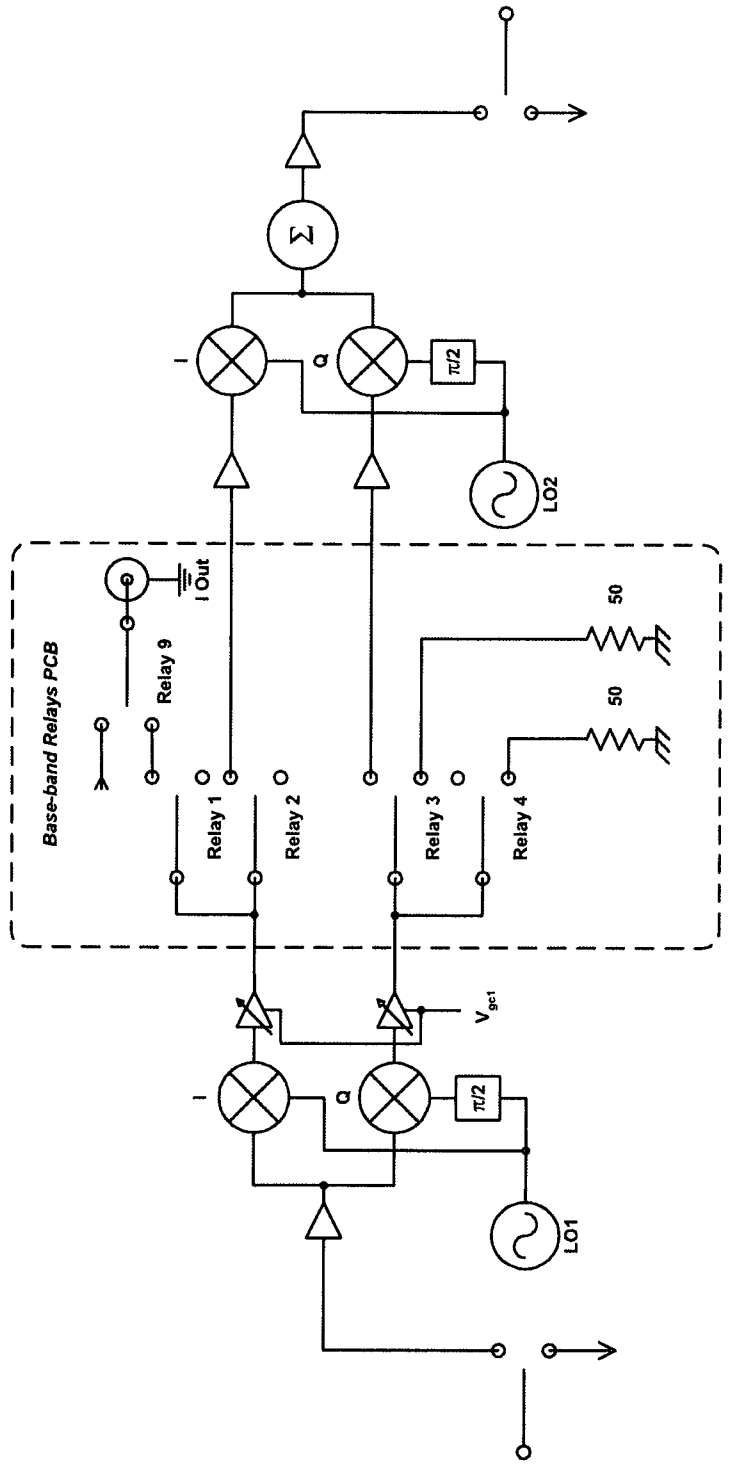


Figure 8.4: Transmit Direction Frequency Converters with Base-band Switching Relays

The third options incur an approximately 3 dB reduction in the signal amplitude seen by the up-converters. This can either be accepted or compensated for by increasing the receiver VGA gains.

The base-band output is only from the in-phase portion of the down-conversion mixers. If the mixers were being used as ISM band zero IF receivers, then this would be a problem. In that instance the received information would be spread across both complex channels as the LO would be arranged to be in the centre of the frequency band of interest. However, here that is not the case. The primary function of the mixers is to change the frequency of the *tester – DUT* interface. Therefore, any LO frequencies that are convenient to that task can be chosen, since it is the difference in the LOs which is important not their absolute values. It is valid therefore, to look only at the in-phase components as the absolute values of the LOs can be chosen to place the base-band information in positive frequencies only. This is fortunate as the high frequency ADC in the MSTS with sufficient depth of capture memory for whole Bluetooth packets has only one input. The equation of the signal presented to the ADC is related to channel number n and downconversion LO frequency $f_{LO}$. If the carrier frequency $f_c = 2402 + n$ MHz then the equation of the signal at the input to the ADC is:

$$\phi(t) = A(t)cos(2\pi(f_c - f_{LO})t + 2\pi f_e(t)t + 2\pi h \int_{-\infty}^{t} m(\tau)\mathrm{d}\tau), \tag{8.3}$$

The choice of in-phase signals for the output is somewhat arbitrary, indeed swapping the connections to the relay PCB would allow the capture of the quadrature components instead. However, the resulting data would be spectrally reversed, which is conceptually harder to consider and would place an additional burden upon subsequent signal processing routines.

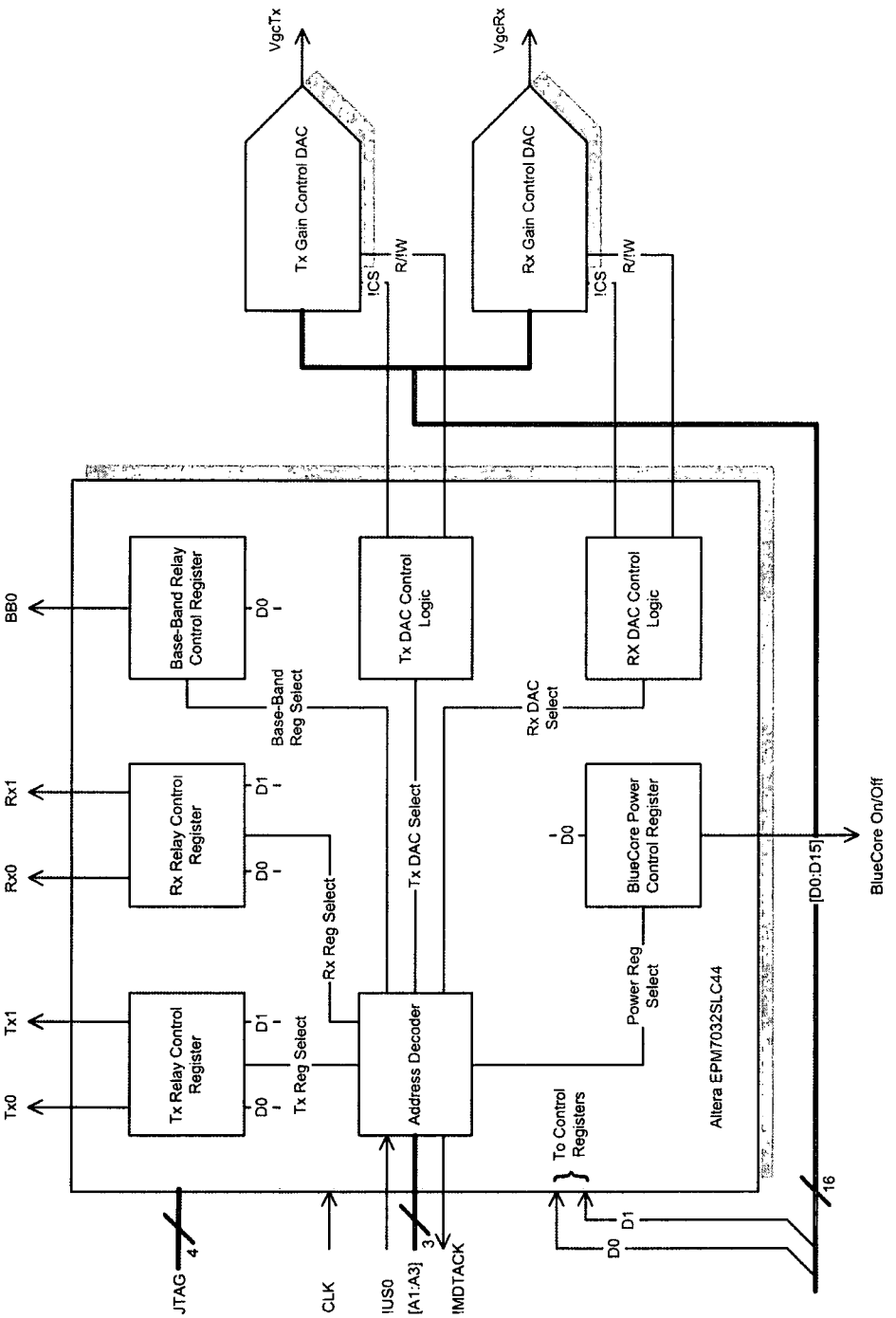


Figure 8.5: Base-Band Control – CPLD Functions and Interface

| Control Signal | | Configuration |
|---|---|---|
| Tx1 | Tx0 | |
| 0 | 0 | No Connections (Power on) |
| 0 | 1 | BlueCore connected to up-converter only |
| 1 | 0 | BlueCore connected to base-band output only |
| 1 | 1 | BlueCore connected to up-converter and base-band output |
| Rx1 | Rx0 | |
| 0 | 0 | No Connections (Power on) |
| 0 | 1 | DUT connected to up-converter only |
| 1 | 0 | DUT connected to base-band output only |
| 1 | 1 | DUT connected to up-converter and base-band output |
| BB0 | | |
| 0 | | Baseband Output from BlueCore |
| 1 | | Baseband Output from DUT |

Table 8.1: Base-Band Relay Switching Table

### 8.2.3 Control Functions

An additional PCB provides supplementary control functions: configuration of the baseband switching relays; programmable gain control voltages for the down-converter VGAs and power on–off reset for the BlueCore. Also there are separate 3.3 V power regulators for each of the four Maxim mixer ICs.

This functionality is largely implemented in an Altera Complex Programmable Logic Device (CPLD), as shown in figure 8.5. Within the CPLD are four data registers, two groups of control logic and an address decoder connected to the least significant user address region of the 68030. The CPLD can therefore be written to from the VXIBus under control of a Labview VI user interface on the host workstation. This VI supervises all control functions described in this section and creates the logical software connection to the tester by obtaining a VISA instrument reference, which is placed in a global variable and used by the other VIs. The internal logic of the CPLD was defined in VHDL, and is reprogrammable via the devices JTAG interface.

Three of the data registers configure the base-band relays in accordance with table 8.2.3. Simply writing the appropriate value to the correct address gives the desired relay setup. The fourth register allows a user to reset the BlueCore by disabling and enabling its voltage regulator output, writing a 1 to the register address enables the regulator. This is the simplest way to close down all Bluetooth links and restart a test.

Down-converter gain control voltages are generated by a pair of 16 bit digital to analogue converters (DACs). When values are written to the DACs the address decoder activates the control logic which goes through a DAC read cycle. Values are indirectly selected by the user who is prompted for a gain value in dB between the up-converter input and down-converter output. A gain for each direction is chosen independently. The correct DAC code is then calculated from an empirical formula and written to the tester. A separate equation is applied in each direction to account for individual device, PCB and cabling losses. The equations were found by connecting the up – down-converter pairs to a VNA and measuring the forward transmission characteristic across the range of valid gain control voltages. Cubic spline functions were then interpolated from these values. It was found that approximately 30 dB of control was achievable, in steps of 0.1 dB , between -35 dB and -5 dB . Loss across the base-band relay board, including cabling and connections, was $< 0.5$ dB .

### 8.2.4 Software Interface

This section describes the BlueCore's host software stack introduced in section 8.2.1. It is based upon CSR's BlueTest application for their Casira development system and has three parts: a Bluetooth protocol stack executable; a Labview VI user interface and a further VI to bridge the protocol stack – VXI – UART gap. Figure 8.6 illustrates this structure.

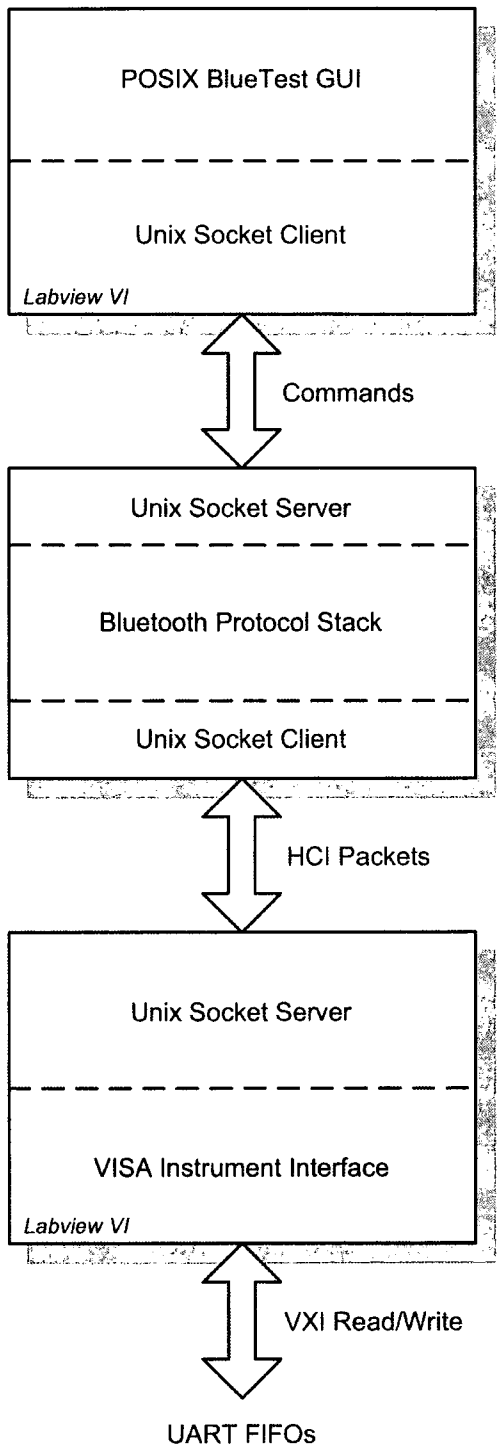


Figure 8.6: BlueCore Host Software Interface – Overall Structure

#### 8.2.4.1 Inter Process Communications

Data flow between the three elements in figure 8.6 is via UNIX sockets. Generally sockets are used to facilitate bidirectional communication between processes on the same machine, inter process communication (IPC), or between physically separate computers over the internet [71]. Here IPC sockets are used as all three of the processes are executing on the same workstation.

Sockets are similar to files; they can be opened, closed, written to and read from, and like files have a numeric, operating system assigned, descriptor. However, there are some important differences. Processes using sockets are able to *listen*[3] for connections from other processes, *accept* such connections and create additional sockets to handle them. This behaviour allows sockets to be the communication end points in the server/client data transfer model. The process listening is the server, whilst the connecting process is the client.

The steps for establishing a server are: obtain an unused socket descriptor and *bind* it to a known location in the file system, `/home/imsadm/socket1` for example; activate the new socket by *listening* for connections; wait for requests from clients which must know the bound location of the listening socket. A client needs to simply obtain its own socket descriptor and attempt to *connect* to the server. When a server receives a connection request it can *accept* it. This creates a new socket connected to the client which is usually passed to a third process that handles the connection, leaving the original server free to deal with new clients.

For the tester's host software the socket related operating system calls have been encapsulated into a dynamic shared library, which is linked into the Bluetooth stack executable and Labview VIs when required. The VIs are therefore able to use sockets to communicate with the Bluetooth stack .

[3]Text in *italics* in section 8.2.4 will refer to a function call to an operating system or library software routine.

#### 8.2.4.2 Customised Bluetooth Host Protocol Stack.

CSR's development kits contain hardware and software to help manufacturers incorporate BlueCore radio modules into consumer electronics products. They also provide a 32 bit Windows application called BlueTest which gives access to the radio self test capabilities of the Casira evaluation hardware connected to a PC.The C++ source code for Bluetest was adapted to be the middle element in figure 8.6. Fortunately BlueTest's protocol stack components were provided ported to the UNIX[4] system and required only organising into a compilable state. Adapting existing software was done to reduce the time required for development of the host stack; writing it from scratch was in no way a practical proposition.

Connections to a chip are therefore handled, ultimately, by the BCSP code. In the original UNIX implementation a connection was opened by a call to *fopen* with the file system path of a UNIX serial port. Normal file IO routines were then used on the returned file descriptor, to send data to and read it from the chip. In the modified code *fopen* is replaced by a call to the *client connect* routine of the IPC shared library, replacing the file path with a socket path and the file descriptor with a socket descriptor. The file IO routines are retained, unmodified, from then on.

The windows BlueTest source code takes the objects of the Bluetooth stack and defines two new classes, *BlueTestEngine* its interface to the protocol stack methods and *BlueTestDlg* which has *BlueTestEngine* as a member and contains the Windows API code. Constructing a *BlueTestDlg* object, which in C++ terms means to load a new instance of the class, an object, into memory, therefore also constructs a *BlueTestEngine* object and so on until a Bluetooth stack is running down to the BCSP layer.

Of course the two BlueTest classes were specific to the Windows environment and so had to be rewritten to remove redundant code and where necessary replace

[4]More accurately it is ported to the POSIX standard architecture, however, there is no documentation with the code and so no comment can be made as to the rigor of its compliance, except to say that i appears to work.

windows specific constructs with their UNIX equivalents. Most of the modifications were required in the *BlueTestDlg* class as it employed callbacks to respond to user input. Callbacks are functions, which execute when a particular action is observed, a left mouse click on an OK button for example. In this implementation there are no callbacks as there is no way for a user to interact directly with the stack. Instead there is a linked list of command data structures which are processed as appropriate. These structures are placed in the list as command strings are received on the upper socket shown in figure 8.6. Similarly responses coming up the stack from the chip are written to this socket. This arrangement gives a simple top-level application, that is the code which uses the *BlueTestDlg* and *BlueTestEngine* classes. It only has three jobs; start the socket server and wait for connections from the GUI VI above; process data strings from the GUI VI and place the resulting command structures in the pending linked list; send pending commands to the Bluetooth stack whilst passing replies up to the GUI. All user interaction with the test master chip is therefore via a Labview interface. This is consistent with IMS's analogue instrumentation philosophy and the design aims stated in section 8.1.1.

#### 8.2.4.3 Host – VXI Interface Server

When the Bluetooth stack constructs its internal objects it attempts to find a chip to connect to. As implemented this causes it to begin issuing connection requests to a socket address. Listening on that socket is the *BlueCore Communications* VI described by figure 8.7. This VI then accepts a connection from the stack and passes control to the *Handle Connection* VI, along with the descriptor of the accepted socket. Host stack to BlueCore communication can now be established. The connection is managed according to the execution diagram in figure 8.8.

Packets, sent as a byte stream from the host BCSP object, are received by the VI and placed in a buffer. Pending data from the BlueCore's BCSP layer are then read from the UART and placed in a second buffer. Any data in the host to BlueCore

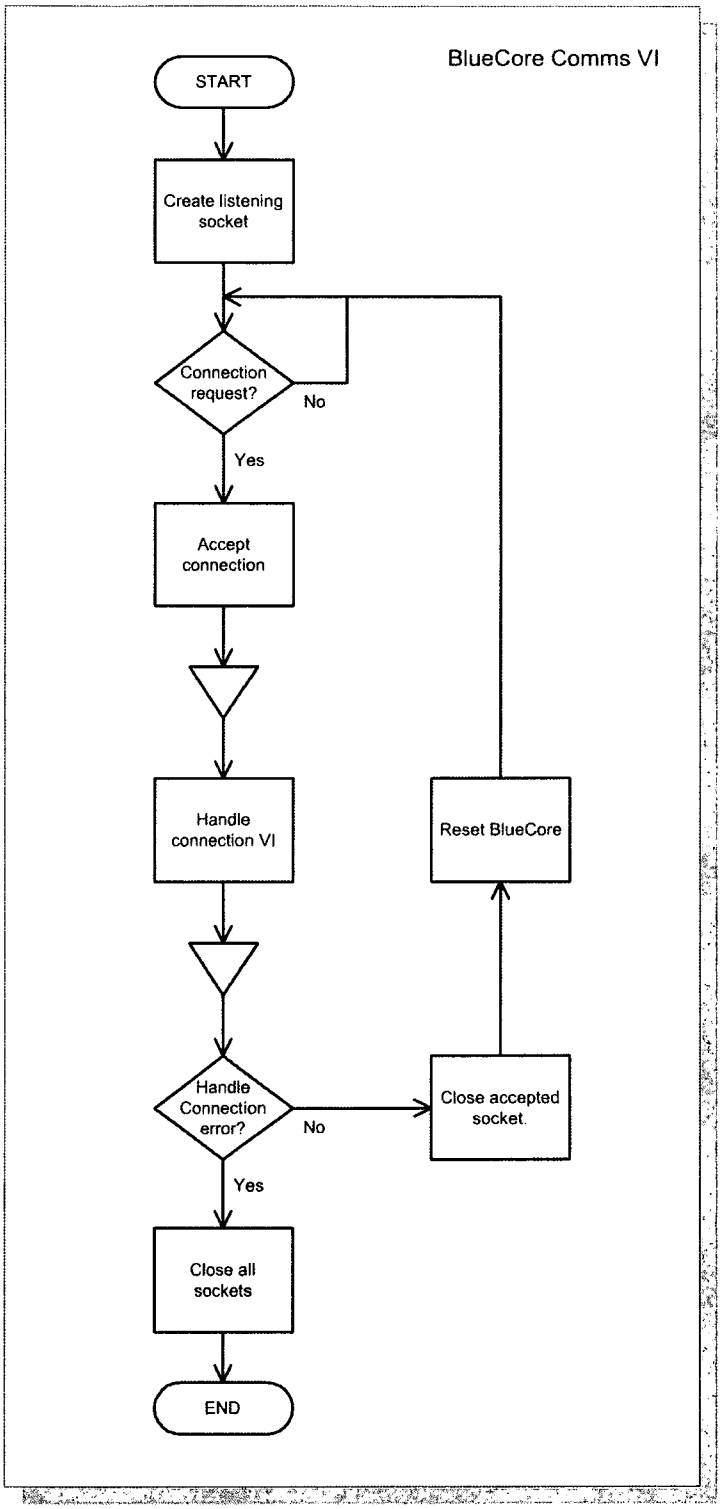


Figure 8.7: BlueCore Communications VI - Execution Diagram

buffer is then written to the UART's receive FIFO until its *FIFO full* flag is asserted. Finally data in the BlueCore to host buffer is written on the socket to the Bluetooth stack, essentially in the form of IPC packets, and the read – write cycle is repeated. If, at any time, an error is encountered, whilst trying to read data from the socket for example, then the *Handle Connection* VI exits and control returns to the *BlueCore Communications* server VI. Errors encountered accessing the accepted socket are ignored by the server; it goes on to accept connections from further instances of the Bluetooth stack. However, a VISA instrument error is assumed to be caused by failure of the host's connection to the VXI mainframe, due to loss of power perhaps, and therefore the server VI also stops. No further communication will be possible with the BlueCore, regardless of whether a healthy host stack is requesting access, until the global VISA reference is successfully updated by the tester control VI (section 8.2.3).

#### 8.2.4.4 Labview Bluetooth Test GUI

This final VI takes the place of the Windows GUI from the original BlueTest application. It duplicates BlueTest's functionality; the same list of commands can be sent to the BlueCore, with the same parameters and default values; responses from the chip are also printed in the VI's front panel as in the Windows version. However, the VI is not part of the Bluetooth protocol executable. Instead, when it runs its first task is to attempt to connect, as a client, to the socket server which forms the application layer of the customised stack software. Figure 8.9 shows the VI front panel.

### 8.2.5 External Instruments

The final elements of the Bluetooth tester are two additional VXI instruments, a Racal 3721 RF source, a fast ADC (see figure 8.1), and a Wiltron 6817B stand alone bench-top RF source, which is the fixed LO for the mixers connected directly to the

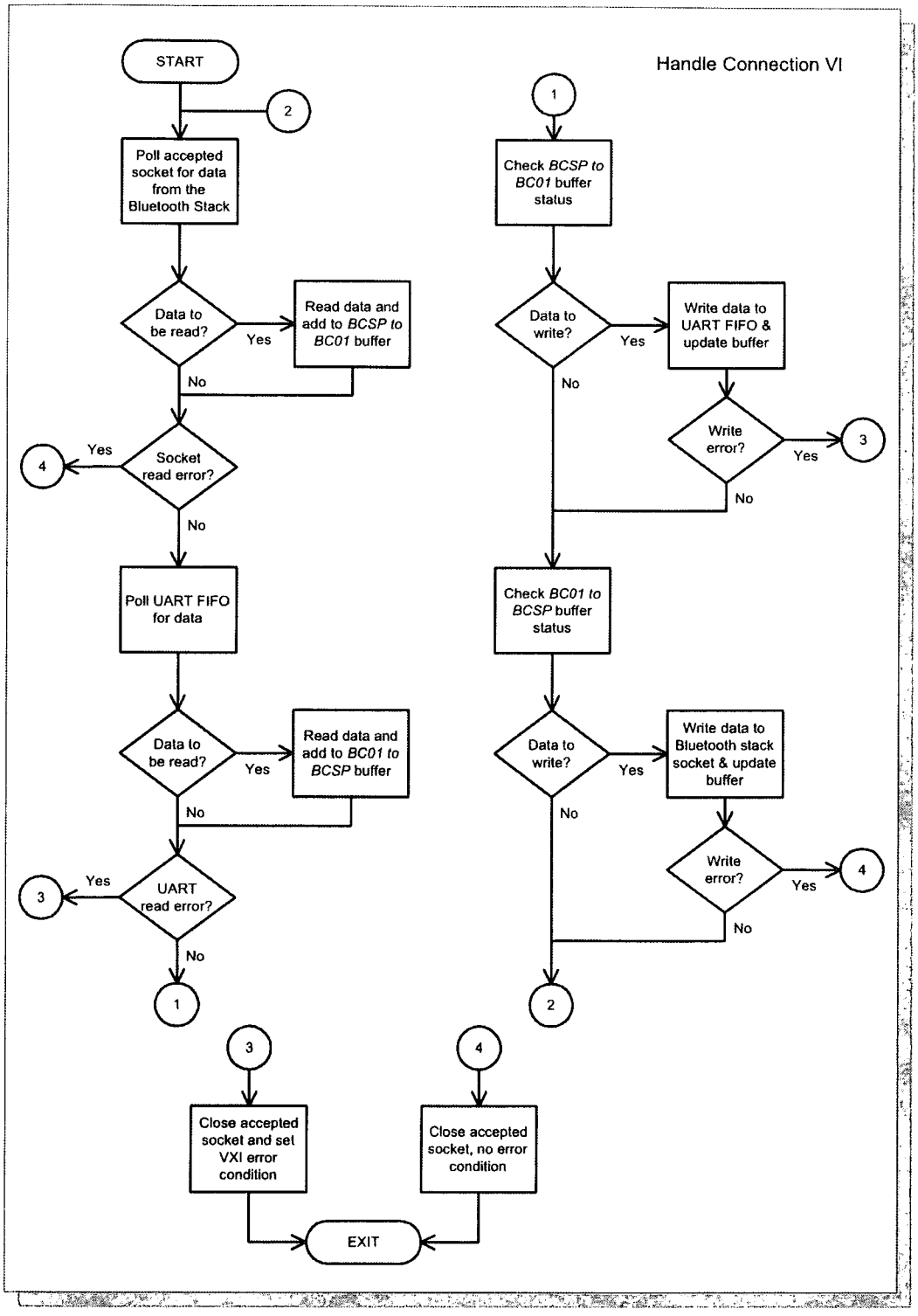


Figure 8.8: Handle Connection VI – Execution Diagram

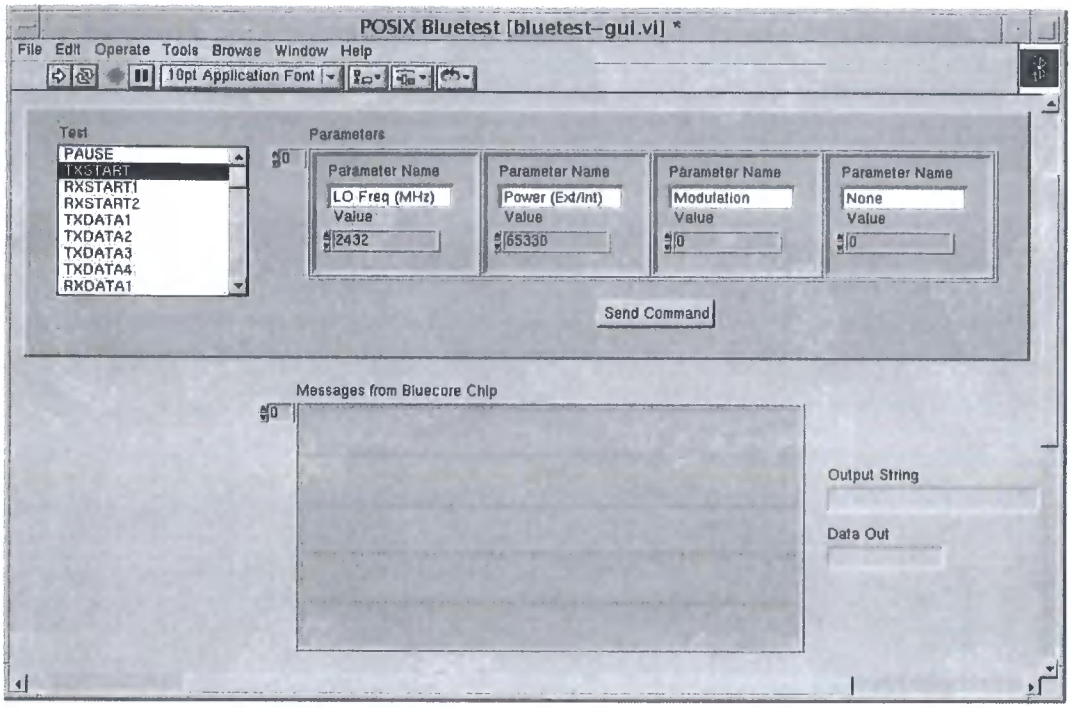


Figure 8.9: Labview BlueTest GUI – Front Panel

BlueCore. The Racal source is the variable LO connected to the mixers on the DUT side and so is used to introduce frequency shifts into the tester's RF output (section 8.2.2). Furthermore, the Racal has a specified frequency resolution of 1 Hz and an accuracy of 10 Hz up to 2.4 GHz.

Capture of base-band packets is achieved with a ZT400VXI, 12 bit, 500 MSample $sec^{-1}$ digitiser from Zetec [72]. It has a sample memory depth of 32 MSamples, sufficient to store twenty two of the longest Bluetooth packets at its full sample rate. Triggering is from the buffered transmit and receive active signals from the BlueCore as appropriate. For instance if DUT packets are desired the receive active trigger is needed. The ZT400VXI's triggering allows the actual capture of data to commence either before or after a trigger event. Therefore, by delaying the capture by the transmitter setup time as specified in [22] sampling can be synchronised to as close to the start of a packet as possible.

## 8.3 Prototype Testing

### 8.3.1 Interface Software

The user interface, host Bluetooth protocol stack and VXI connection server have been implemented as described. All three elements have bee shown to work together. The BlueCore Communication server VI accepts connections from the host stack executable and successfully hands over control to the Handle Connection VI. The IPC socket server at the top level of the Bluetooth stack also accepts a connection from the Labview GUI correctly as part of its object initialisation procedure. Commands can be assembled in the GUI and are received correctly by the protocol stack. Replies from the stack software are received by the GUI and displayed as messages validating the Windows callback replacement mechanism.

Overall, the software interface has shown that it is possible to run a complex communications protocol stack within a Labview environment.

### 8.3.2 Hardware Tests

Testing of the VXI module hardware gave promising results. The control electronics worked correctly. A connection was established between Labview and the FPGA in the VXI mainframe allowing baseband signal paths to be configured via the relay PCB.

Unfortunately it was not possible to rectify faults within the BlueCores PCB. Therefore a complete link between the host workstation and master Bluetooth device was not formed. However, Bluetooth packets were captured by down converting output from an Anritsu Bluetooth test set with a MAXIM 2701. The sampled data is shown in figure 8.10 and detail from the start of the packet is shown in figure 8.11. The signal is an ID packet which consists only of an access code (see section 3.2) and is used for device discovery before a link is established. The local oscillator frequency used was 2.4 GHz generated by the Racal 3721 in the MSTS's VXI mainframe. The

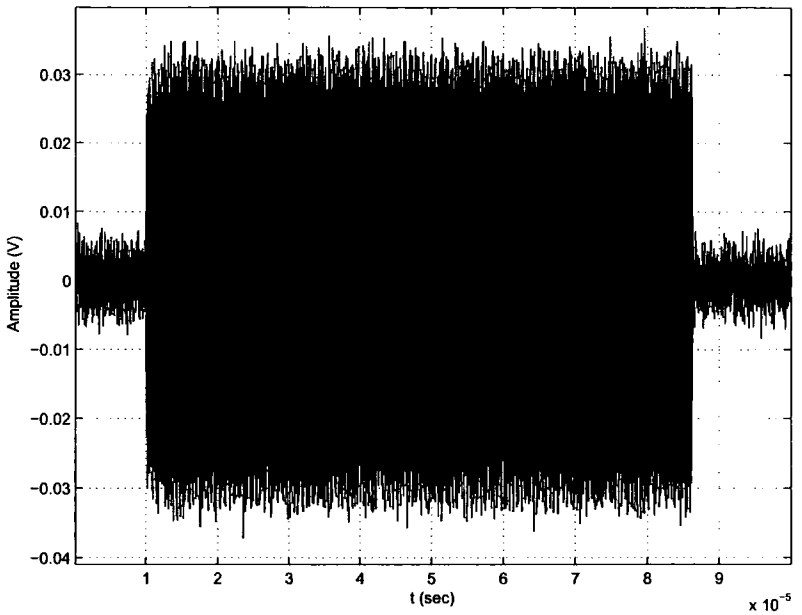


Figure 8.10: Bluetooth ID Packet at Baseband (LO = 2.4 GHz, $f_s$ = 250 Msample/sec.)

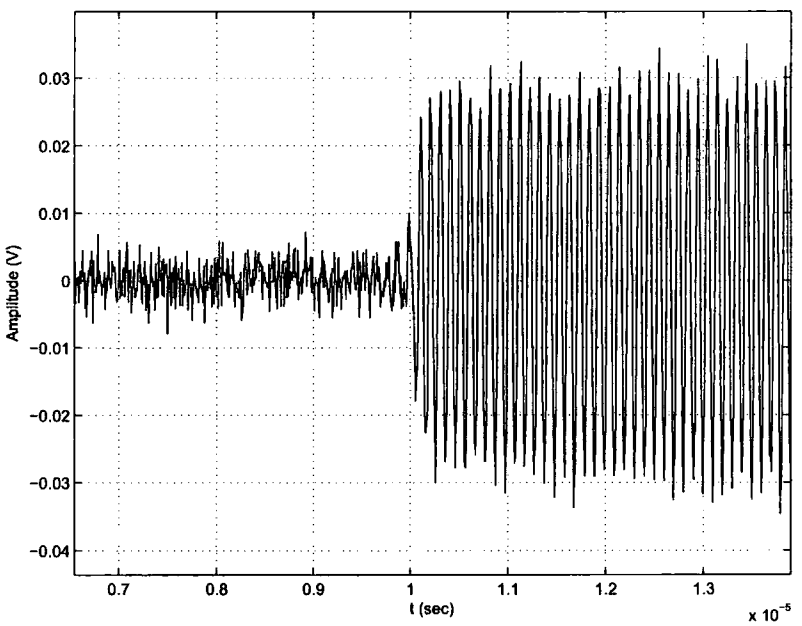


Figure 8.11: Detail of ID Packet Start (LO = 2.4 GHz, $f_s$ = 250 Msample/sec.)

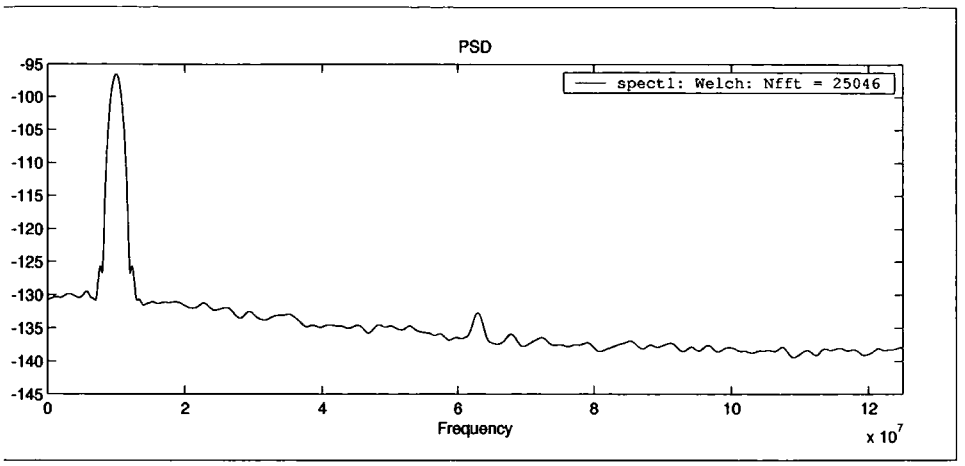


Figure 8.12: Spectrum of ID Packet: GFSK Signal at 10 MHz (LO = 2.4 GHz, $f_s$ = 250 Msample/sec.)

spectrum of the sampled data, figure 8.12, reveals that the original frequency of the signal was 2.410 GHz as the peak spectral component occurs at 10 MHz. With this knowledge it was possible to demodulate the captured signal with the Phase Shift Discriminator Matlab routines developed for the simulations described in earlier chapters. Figure 8.13 illustrates the output of the demodulator. The packet's '1010' preamble can be seen starting at approximately 15 µs. There is no access code postamble as a payload header does not follow on in an ID packet.

Capturing a Bluetooth packet verified that the baseband signal access part of the instrument's design is valid. It also provides conformation that the Matlab demodulation software works correctly.

## 8.4 Conclusion

The development of the Bluetooth test module provides a starting point for a more comprehensive WPAN radio verification system. The frequency shifting and baseband signal analysis capability could be used for any protocol operating in the 2.4 GHz ISM band, for instance with the first ZigBee devices which are now coming into production. Testing new protocols would require different RF electronics and host

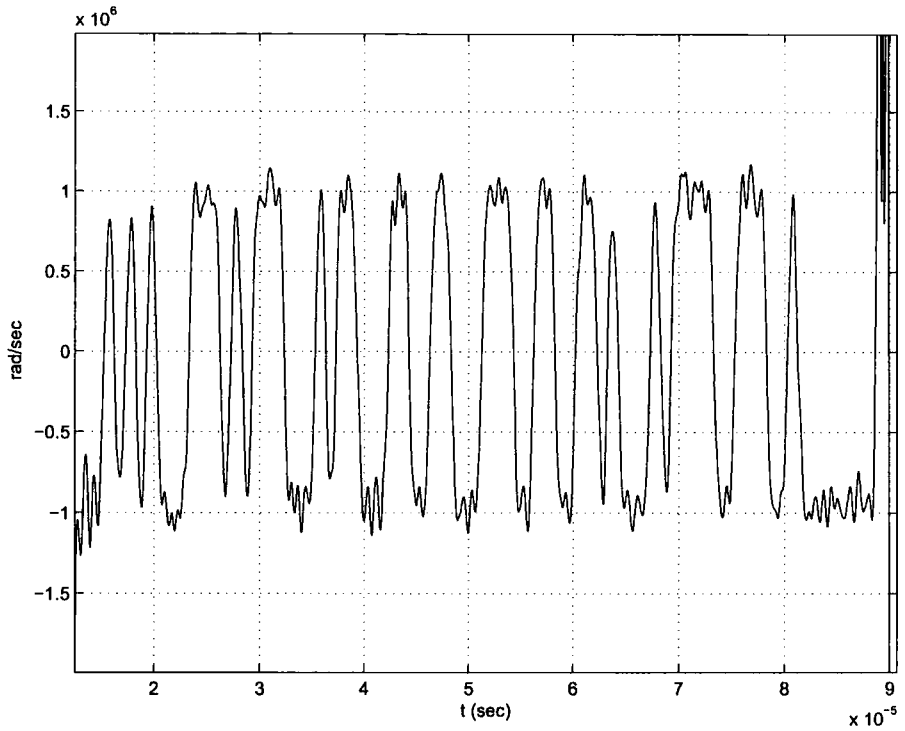


Figure 8.13: Demodulated ID Packet, [1010] Preamble Bit Pattern at 15 μs.

software, but the mechanism for controlling a software stack through Labview has been proven. Testing of future devices, which use wireless RF for inter-chip communications, will need such an arrangement. Test equipment will have to be able to generate signals conforming to whatever standard is used. An ability to employ a variety of protocol implementations within a unified framework such as Labview will be significant in this context.

# Chapter 9

# Conclusions

As the sales of Bluetooth chip sets passes 1 million per week, WLAN usage expands and ZigBee products become available, the noise in the limited spectrum in which they all operate will increase. Therefore, it is important that each radio system uses bandwidth in the most efficient way possible, consistent with the demands of a user's application. One way to ensure this is to optimise the signal to noise ratio required by receivers for error free communication. This thesis has highlighted an aspect of the Bluetooth system which can render it less than optimally efficient and offered mechanisms to both quantify and compensate for this effect.

The Bluetooth specification is extremely comprehensive. In the core document of the latest version there are 1200 pages and implementations must demonstrate compliance before they can be released into the market place. Nevertheless, strict adherence to the specification is not a sufficient criterion for a good radio design. Simulations have shown that when GFSK modulated signals with carrier frequency errors in the range allowed by the physical layer radio specification are demodulated, the bit error rate performance of a receiver declines significantly. In the case of a 70 kHz transmitted initial frequency error the SNR required to achieve the minimum BER of $10^{-3}$ almost doubled to 30 dB. This situation degrades further if both receiver and transmitter experience carrier frequency errors. The mutual frequency

mismatch can become greater than the frequency deviations used to encode data, rendering communication impossible regardless of SNR.

Measuring the carrier frequency characteristics of Bluetooth radios is therefore essential. An algorithm to do this was developed and tested in conjunction with Advantest, a manufacturer of automatic test equipment for IC production. An ability to estimate carrier frequency errors was demonstrated and the accuracy of the method was shown to be close to the theoretical limit expressed as a lower bound on the estimate's standard deviations. Furthermore, the algorithm was integrated into the software environment of a standard RF IC test system, providing a means to test Bluetooth transceivers without recourse to special communications device analysis equipment. Advantest consider this to be advantageous in terms of their equipment's flexibility and the reduced cost of testing ICs that this implies.

Since the performance of Bluetooth receivers is affected by carrier frequency errors a method of compensating for them is desirable. The method of compensation should be as simple as possible so as not to adversely affect the complexity and cost of a radio design. Ultimately the elevated error rates are caused by a DC offset being introduced into the demodulated signal. Since, after demodulation, data bits are recovered by a decision algorithm it is logical to alter its operation to account for the offset. This provides the simplest solution. Consequently, an Adaptive Threshold Bit Slicer decision algorithm was proposed. It functions by estimating the frequency error induced offset during a packet's preamble. This estimate is then used to adapt the decision threshold in the subsequent data recovery stage. The efficacy of the method was demonstrated through simulations. Even in the most severe cases additional BERs due to carrier frequency errors were almost completely eliminated. A Bluetooth receiver implementing this compensation method would have superior BER performance. This translates to better throughput, a higher quality of service and a more efficient use of available radio spectrum.

A Bluetooth test system for engineering test and verification was developed. It

was designed to enhance the capabilities of the University's Mixed Signal Test Station and was based upon the VXIBus form factor. The user interface was through National Instrument's Labview, demonstrating that complex communication protocol software can be controlled from within a separate, unified test environment. An ability to capture Bluetooth signals at base band frequencies was also demonstrated and used to verify GFSK demodulation routines used in earlier simulations.

Bluetooth is the most established of the emerging WPAN standards. Whether or not WPANs will become as significant as the cellular telephone or the internet remains to be seen. However, they are an enabling technology of what is widely believed to be the computing paradigm of the future: Ambient Intelligence.

# Bibliography

[1] Office of Telecommunications. (2003, Apr.) Consumers use of mobile telephony Oftel residential survey. [Online]. Available: http://www.oftel.gov.uk/publications/research/2003/q12mobr0403.pdf

[2] T. S. Rappaport, *Wireless Communications Principles and Practice*, 2nd ed. Upper Saddle River, NJ 07458: Prentice Hall, 2002.

[3] E. J. Cooper-Green. (2003, Apr.) Internet Access: Individuals and Households. [Online]. Available: http://www.statistics.gov.uk/pdfdir/int0703.pdf

[4] K. J. Negus, A. P. Stephens, and J. Lansford, "HomeRF: Wireless Networking for the Connected Home," *IEEE Personal Comunications*, vol. 7, pp. 20–27, Feb 2000.

[5] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman, "Scenarios for Ambient Intelligence in 2010," in *IPTS*, Seville, Feb. 2001.

[6] M. Weiser. (1996) Ubiquitous computing. [Online]. Available: http://www.ubiq.com/hypertext/weiser/UbiHome.html

[7] N. Shadbolt, "Ambient Intelligence," *IEEE Intelligent Systems*, vol. 18, pp. 2–3, July/August 2003.

[8] D. Estrin, D. Culler, K. Pister, and G. Shukhatme, "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, vol. 2, pp. 59–69, January/March 2002.

[9] V. P. R. M. C. F. Chang and, L. Zhang, H. Shin, and Y. Qian, "RF/Wireless Interconnect for Inter- and Intra-Chip Communications," *Proceedings of the IEEE*, vol. 89, pp. 456–466, April 2001.

[10] R. Haverman and J. A. Hutchby, "High-Performance Interconnects: An Integration Overview," *Proceedings of the IEEE*, vol. 89, pp. 586–601, May 2001.

[11] ITRS. International Technology Roadmap for Silicon 2001, Interconnect. [Online]. Available: http://public.itrs.net

[12] A. Sivard, "Implantable RF medical devices: Designing for ultra-low power," *Wireless Design and Development*, November 2003.

[13] C. Ekland, R. B. Marks, K. L. Stanwood, and S. Wang, "Ieee Standard 802.16: A Technical Overview of the WirelssMAN Air Iinterface for Broadband Wireless Access," *IEEE Communications Magazine*, vol. 40, pp. 98–107, June 2002.

[14] ETSI. Broadband Radio Access Networks: Brochure. [Online]. Available: http://www.etsi.org/literature/HFAbrochures/pdf/!branpro.pdf

[15] W. I. for Microwave Access Forum. WiMAX Overview. [Online]. Available: http://www.wimaxforum.org/news/downloads/WiMAX-Overview.pdf

[16] S. J. Kerry. IEEE 802.11 WG activities:. [Online]. Available: http://grouper.ieee.org/groups/802/11/

[17] ETSI. (1999, Apr.) Bran: Hiperlan type 1, functional specification, en 300 652 v1.2.1. [Online]. Available: http://portal.etsi.org/bran/kta/Hiperlan/hiperlan1.asp

[18] ——. (2000, Apr.) Bran: Hiperlan type 2, Physical Layer, etsi ts 101 475 v1.1.1. [Online]. Available: http://portal.etsi.org/bran/kta/Hiperlan/hiperlan2.asp

[19] R.Shim. (2003, Jan.) CNET News: Home RF Working Group Disbands. [Online]. Available: http://www.news.com

[20] T. M. Siep, I. C. Gifford, R. C. Braley, and R. F. Heile, "Paving the Way for Personal Area Network Standards: An Overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks," *IEEE Personal Communications*, vol. 7, pp. 37–43, Feb 2000.

[21] I. D. Association. (2001, May) Infrared Data Association Serial Infrared Physical Layer Specification: Version 1.4. [Online]. Available: http://www.irda.org/standards/pubs/IrPHY1p4.pdf

[22] Bluetooth SIG. (2003, Jan.) Bluetooth V1.1 Core Specification. [Online]. Available: http://www.bluetooth.com/dev/specifications.asp

[23] V. Bahl. (2002) Zigbee overview. [Online]. Available: http://www.zigbee.org/documents/ZigBeeOverview4.pdf

[24] J. Bray and C. F. Sturman, *Bluetooth: Connect Without Cables*, 1st ed. Upper Saddle River, NJ 07458: Prentice Hall, 2001.

[25] R. Schiphorst, F. Hoeksema, and K. Slump, "Bluetooth Demodulation Algorithms and their Performance," in *Proc. 2nd Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, Mar. 2002, pp. 99–106.

[26] C. S. Radio. (2004, Feb.) CSR and Silicon Wave collaborate on testing medium-rate Bluetooth technology. [Online]. Available: http://www.csr.com/pr/pr163.htm

[27] J. Putscher, "Bluetooth 2003: Are PMGs Another Driver?"

[28] A. Research. (2003) Automotive wireless networks. [Online]. Available: http://www.abiresearch.com/reports/AWN.html

[29] C. E. Sunderg, "Continuous Phase Modulation," *IEEE Communications Magazine*, vol. 24, pp. 25–38, April 1986.

[30] Ericsson Tachnology Licensing AB. (2003, Aug.) Bluetooth Beginners Guide. [Online]. Available: {http://www.ericsson.com/bluetooth/beginners-files/beginners-guide.pdf}

[31] A. El-Hoiydi, "Interference Between Bluetooth Networks - Upper Bound on the Packet Error Rate," *IEEE Communications Letters*, vol. 5, pp. 245–247, June 2001.

[32] A. Karnik and A. Kumar, "Performance Analysis of the Bluetooth Physical Layer," Dec. 2000.

[33] T.-Y. Lin and Y.-C. Tseng, "Collision analysis for a multi-bluetooth picocells," *IEEE Transactions on Communications*, to appear. [Online]. Available: citeseer.nj.nec.com/599413.html

[34] Bluecore Features: CQDDR. [Online]. Available: http://www.csr.com/products/CQDDR.htm

[35] Bluetooth SIG. (2003, Feb.) Bluetooth RF Test Specification 1.1. [Online]. Available: https://www.bluetooth.org/admin/bluetooth2/test/index.php

[36] R. de Buda, "Coherent Demodulation of Frequency Shift Keying With Low Deviation Ratio," *IEEE Transactions on Communications*, vol. 20, pp. 429–435, June 1972.

[37] H. Nyquist., "Certain Topics in Telegraph Transmissions Theory," *Transactions of the AIEE*, vol. 47, pp. 617–644, Feb 1928.

[38] F. G. Stremler, *Introduction to Communications Systems*, 3rd ed. Reading, Massachusetts: Addison Wesley, 1992.

[39] K. Murota and K. Hirade, "GMSK Modulation for Digital Mobile Telephony," *IEEE Transactions on Communications*, vol. 29, pp. 1044–1050, Nov 1981.

[40] R. E. Ziemer, *Introduction to Digital Communication*, 1st ed. New York, NY 10022: Maxwell Macmillan, 1992.

[41] J. Sebesta and M. Kasal, "Digital Costas Loop Applications in FSK Demodulator," in *Proceedings of the 10th Aachen Symposium on Signal Theory*, Aachen, Germany, September 2001, pp. 435–438.

[42] J. C. Haartsen., "The Bluetooth Radio System," *IEEE Personal Communications*, pp. 28–36, Feb 2000.

[43] M. Shimizu and et. al., "New Method of Analyzing Performance of GFSK with Post Detection Filtering," *IEEE Transactions on Communications*, vol. 45, pp. 429–436, April 1997.

[44] H. Darabi and et. al., "A 2.4GHz CMOS Transceiver for Bluetooth," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 2016–2023, Dec 2001.

[45] J. C. Ng, K. B. Letaief, and R. Murch, "Performance of Adaptive DFE and Antenna Modems for Wireless Personal Communications," in *Proceedings of the IEEE International Conference on Universal Personal Communications*, Florence, Italy, October 1998.

[46] W. Chen. (2001, Dec.) AN2211/D Motorola's Bluetooth Solution to Interference Rejection and Coexistance with 802.11. [Online]. Available: http://e-www.motorola.com/brdata/PDFDB/docs/AN2211.pdf

[47] A. Soltanian and R. E. V. Dyck, "Performance of the Bluetooth System in Fading Dispersive Channels and Interference," in *Proceedings of IEEE Globecom*, San Antonio, Texas, Nov 2001.

[48] W. Zhang and M. J. Miller, "Baseband Equivalents in Digital Communication System Simulation," *IEEE Transactions on Education*, vol. 35, pp. 376–382, Nov 1992.

[49] T. Newton, Camridge Silicon Radio, August 2003, Personal Communication.

[50] Rohde and Schwarz GmbH and Co. (2003, Aug) Universal Radio Communication Tester CMU200. [Online]. Available: http://www.rohde-schwarz.com/

[51] Anritsu. (2001, Dec.) Bluetooth test solutions. [Online]. Available: http://www.us.anritsu.com/downloads/files/Bluetoothdatasheet02.pdf

[52] E. O. Brigham, *The Fast Fourier Transform and its Applications.* Englewood Cliffs, New Jersey, edition =: Prentice Hall International.

[53] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, 1st ed. Englewood Cliffs, NJ: Prentice Hall, 1975.

[54] S. L. Marple, *Digital Spectral Analysis: With Applications.* Englewood Cliffs, NJ: Prentice Hall, 1987.

[55] Mathworks - Matlab. (2006) Spectral Estimation Methods :: Statistical Signal Processing. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/signal/spectra6.html

[56] M. Banerjee. (2003, Aug) Generating Bluetooth Signals with SMIQ and the Application Software SMIQ-K5. [Online]. Available: http://www.rohde-schwarz.com/

[57] A. J. Fisher. (2003, Aug) Interactive Digital Filter Design. [Online]. Available: http://www-users.cs.york.ac.uk/~fisher/mkfilter/

[58] K. Itoh, "Analysis of the Phase Unwrapping Algorithm," *Applied Optics*, vol. 21, p. 2470, July 1982.

[59] D. C. Chiglia and M. D. Pritt, *Two-Dimensional Phase Unwrapping Theory, Algorithms and Software*, 1st ed. New York: Wiley Inter-Science, 1998.

[60] B. Boashash, "Estimating and Interpreting the Instantaneous Frequency of a Signal - Part II: Algorithms and Applications," *Proceedings of the IEEE*, vol. 80, pp. 540–568, April 1992.

[61] S. Holm, "Optimum FFT-Based Frequency Acquisition with Application to COPSAS-SARSAT," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, pp. 464–475, April 1993.

[62] D. C. Rife and R. R. Boorstyn, "Single Tone Parameter Estimations from Discrete-Time Observations," *IEEE Transactions on Information Theory*, vol. 20, pp. 591–598, September 1974.

[63] E. Kreyszig, *Advanced Engineering Mathematics*, 7th ed. John Wiley and Sons Inc, 1993.

[64] A. N. D'Andrea, U. Mengali, and R. Reggiannini, "The Modified Cramer-Rao Bound and its Application to Synchronisation Problems." *IEEE Transactions on Communications*, vol. 42, pp. 1391–1399, Feb/Mar/Apr 1994.

[65] V. Abhayawardhana and I. Wassell, "Residual frequency offset correction for coherently modulated OFDM systems in wireless communications," 2002. [Online]. Available: citeseer.nj.nec.com/abhayawardhana02residual.html

[66] S. Hong and Y. H. Lee, "Fractionally-Spaced Differential Detection of GFSK Signals with Small h," *IEICE Transactions on Communication*, vol. E84-B, pp. 3226–3234, Dec 2001.

[67] J. Notor, "CMOS Bluetooth RFIC Design Considerations," Wireless Design and Development, August 2001. [Online]. Available: http://www.wirelessdesignmag.com

[68] J. Collier, Camridge Silicon Radio, November 2001, E-Mail Communication.

[69] NEC. (1999) GaAs Integrated Circuit UPG153TB Data Sheet. [Online]. Available: http://www.ncsd.necel.com/microwave/index.html

[70] (2001) BlueCore01b Single Chip Bluetooth Device. [Online]. Available: http://www.csrsupport.com/public/24_bc01b-ds-003d.pdf

[71] C. Brown, *UNIX Distributed Programming*, 1st ed. Hemel Hempstead UK: Prentice Hall, 1994.

[72] ZTEC. (2004) Zt400vxi 500 ms/s, 12-bit digital storage oscilloscope. [Online]. Available: http://www.ztec-inc.com/downloads/products/datasheets/ZT400.pdf

# Appendix A

# Matlab Simulation Source Code

The following source code extracts are taken from the Matlab simulations described in chapters 5 and 7. Matlab syntax is used throughout; lines which begin with a % symbol are comments.

## A.1 Transmitter Code

The top level transmitter function `bt_mod_offset_drift` creates a Bluetooth modulated signal on channel `chan`, with system sampling frequency fs Hz and containing `len` symbols (bits). The bit pattern is created by a uniform random process (`randsrc`). The resultant signal is GFSK modulated on a carrier frequency of `chan + offset + 2` MHz and pre-modulation filtered with a Gaussian filter. `mod_index` may be omitted in which case default value of 0.315 will be used [22]. `offset` is the carrier frequency offset in Hz, it may be positive or negative and may be omitted, eg 1 MHz will cause signal to be on channel `chan + 1`. The first argument controls the generation of input data: `[200,0]` will generate a new data set of 200 symbols, `[200,1]` or just `[200]`, will generate no new symbols unless the global variable containing previous data sets are empty. The returned values are the transmitted data and a vector of sampled GFSK waveform.

```
function [data,mod_sig] = bt_mod_offset_drift(len,
                                              chan,
                                              fs,
                                              mod_index,
                                              offset,
                                              drift);

global x_filt; global data; global T;
% Clear global x_filt and generate new data if requested.
[r,c] = size(len); if c > 1
    if len(2) == 0
        clear global;
        global x_filt;
        global data;
        global T;
    end
end

if(nargin < 3)||(nargin>6)
    error('bt_mod: Wrong number of arguments.');
end if nargin < 6
    drift = 0;
end if(nargin <5)
    offset = 0;
end if(nargin < 4)
    mod_index = 0.315;
end

% Only generate new data when required
% ie after a clear above or on first run.
if size(x_filt) == 0

    x = randsrc(len(1),1,[-1,1]);
    % x is column vector of len elements of either
    % 1 or -1 with equal probability. Each entry
    % represents a 1 usec duration symbol.

    if x(1) == 1
        preamble = [1,-1,1,-1];
    else
        preamble = [-1,1,-1,1];
    end
    % Use appropriate preamble.

    x = [preamble,x']; % Prepend preamble to data vector.
    x = x';
```

```
    T = (1/fs:1/fs:((len1)*1e-6))';
    % t is a time vector of sample points 1/fs
    % apart and len+4 usecs in
    % duration.

    x_samp = expand_seq(x,fs);
    % x_samp is sampled input data.
    % fs is divided to give a value in MHz as the
    % expand_seq function assumes that the data in
    % its input vector consists of symbols of
    % 1 usec duration.

    b = gaussianpulsefilter(640,500e3,1e-6,2);
    % Gives correct response for a sampling frequency
    % of 320MHz and truncates the filter to 2 bit periods.

    x_filt = filter(b,1,x_samp);
    x_filt = x_filt./max(x_filt);
    % x_filt is the normalised and
    % filtered version of x_samp.
end

mod_sig = bt_vco_offset_drift(x_filt,
                              chan,
                              fs,
                              mod_index,
                              offset,
                              drift);
% mod_sig is the final BT modulated signal.
```

The function `randsrc` is a Matlab routine. Initial sampling of the packet data at `fs` is performed by `expand_seq`. In order to ensure that there are a whole number of samples per symbol `fs` must be an integer number of MHz and greater than 2 MHz to satisfy Nyquist.

```
function out = expand_seq(in,fs)

if (fs<2e6) || (floor(fs/1e6) ~= fs/1e6)
    error('fs must be at least 2 (MHz)
    and an integer multiple of 1MHz.');
end

n = length(in);
```

```
% n = length of input data in usec.

samp_per_sym = floor(fs/1e6);%
plus_samples = ones(1,samp_per_sym);%
minus_samples = -plus_samples;%

out =[];%
t=1;%
while t <= n
    if in(t) == 1
        out = [out,plus_samples];
    else
        out = [out,minus_samples];
    end
    t=t+1;
end
```

Gaussian premodulation filtering is performed by the Matlab `filter` routine with coefficients generated by the `gaussianpulsefilter` function which has four parameters. The first, `N` is the order of the filter, `B` is the 3 dB bandwidth of the filter, `T` is the symbol duration and `P` is the number of periods over which the the filter's response is calculated. Referring to figure 4.2 it can be seen that truncation after two periods is satisfactory for $BT = 0.5$ as in Bluetooth [47]. A filter for GSM GMSK, which has $BT = 0.3$ would require three periods. For a system sampling frequency of 320 MHz, recommended values of N to give a -3 dB point at a normalised frequency of $3.125e^{-3}$ ($B = 500$ kHz) are 640 with $P = 2$ and 956 with $P = 3$.

```
function h = gaussianpulsefilter(N,B,T,P)

if B*T < 0.3
    warning('BT <0.3: filter may be too short.');
end

if nargin < 4
    if B*T >= 0.5
        P = 3;
    else
        P = 2;
    end
```

```
else
    if (P~= 2) && (P ~= 3)
        error('P must be 2 or 3');
    end
end

% Truncate filter response:
if P == 2
    n = linspace(-T,T,N);
else
    n = linspace((-3*T)/2,(3*T)/2,N);
end

% Equation for impulse response of the filter from [1]
h = (B*sqrt(2*pi/log(2)))*exp(-(2*pi^2*B^2*n.^2/log(2)));
% Scale the resultant samples of the Gaussian
h = h./sum(h);

% If no output variable is specified
% display the filter response in
% fvtool.
if nargout == 0,
    fvtool(h,1);
end
```

FM modulation of the gaussian filtered data is done by the `bt_vco_offset_drift` function. It is based upon the Matlab vco and `modulate` routines with appropriate modifications to accommodate carrier frequency errors. The last three arguments, `mod_index`, `offset` and `drift` may be omitted, in which case default values of 0.315, 0 Hz and 0 Hz respectively will be used.

```
function out = bt_vco_offset_drift(in,
                                   channel_number,
                                   fs,
                                   mod_index,
                                   offset,
                                   drift)

if (nargin < 3) || (nargin > 6)
    error('bt_vco: Wrong number of arguments.');
end if(channel_number > 78)
    error('Use channel numbers between 0 and 78.')
end
```

```
if nargin < 6
    drift = 0 ;
end

if nargin < 5
    offset = 0;
end

if(nargin < 4)
    dev = 157.5e3;
    % 157.5KHz deviation -> 0.315 if non specified.
else
    dev = mod_index/(2*1e-6);
end

[r,c]=size(in);
if (r==1),
% convert row vector to column.
    in = in(:);
    len = c;
else
    len = r;
end

in_max = max(max(in)); in_min = min(min(in)); if
(in_max>1)|(in_min<-1)
    error('   input vector outside of range [-1,1]')
end

% Modulate to 2MHz plus channel number +- deviation + offset(Hz).

%fmin = 2e6 + (channel_number * 1e6) - dev;
%fmax = 2e6 + (channel_number * 1e6) + dev;

kf = (dev/fs)*2*pi;
% Constant of freq modulation.
fc = 2e6 + (channel_number * 1e6) + offset;
% vco.m uses fc = mean([fmin,fmax]);

% Relevant lines from modulate.m:
t = (0:1/fs:((len-1)/fs))';
%fm modulate input onto fc.
out = cos(2*pi*fc*t + 2*pi*drift*t.*t + kf*cumsum(in));
```

## A.2 Receiver Code

### A.2.1 Quadrature Detector

The following function, `fm_discrim`, demodulates a GFSK signal according to the quadrature discrimination method described in section 4.2.1.1.

```
function [nrzgauss,IFFilt,x] = fm_discrim(mod_sig,chan,fs,n);

if nargin >3
    if (floor(n/2) == n/2) || (floor(n) ~= n)
        error('fm_discrim(mod_sig,chan,fs,n): n must be an odd integ
    else
        delay = n*fs/(4*2e6);
    end
else
    % Use minimum delay (40 samples for fs = 320MHz.)
    n = fs/(4*2e6);
end

len  = length(mod_sig); t  = 0:1/fs:(len-1)/fs;

% Design low pass filter:
% Pass band freq (normalised to fs/2).
Wp = 2*1e6/fs;
% Stop band freq (normalised to fs/2).
Ws = 2*4e6/fs;
% 1dB ripple in passband and 80dB atten at Ws.
[n,Wn] = cheb2ord(Wp,Ws,1,65);
% Chebyshev Type II IIR filter coefficients.
[b,a] = cheby2(n,65,Wn);

% Flo is difference between channel frequency and IF frequency:
if chan ~= 0
    Flo = chan*1e6;
    LO = cos(2*pi*Flo*t)';
    IF = mod_sig.*LO;
    IFFilt = filter(b,a,IF);
else
    IFFilt = mod_sig';
end

padding = zeros(1,delay);
IFFiltPad = [IFFilt,padding];
```

```
IFFiltDelay = [padding,IFFilt];

xlong = IFFiltPad.*IFFiltDelay;

% Remove extra values from the end of xlong.
x = xlong(1,1:length(IFFilt));

% Use same filter as before.
nrzgauss = filter(b,a,x);
```

### A.2.2 Phase Shift Discriminator

The following function, `phasediscriminator`, and its subroutines demodulate a GFSK signal according to the phase shift discriminator method described in section 4.2.1.2.

```
function nrzgauss = phasediscriminator(mod_sig,chan,fs,diff)
% Stage 1. Downconvert mod_sig to baseband.

[Ibb,Qbb] = mod_sig_downconv(mod_sig,chan,fs);

% Stage 2. Low pass filter baseband signals

% Use Butterworth filter (type = 0).
[IbbFilt,QbbFilt] = pd_lpf(Ibb,Qbb,fs,0);

% Stage 3. Extract phase

phase = phase_extract(IbbFilt,QbbFilt);

% Stage 4. Differenciate phase to retrieve NRZ message.

if (nargin < 4)
    nrzgauss = phase_to_nrz(phase,1);
else
    if diff == 0
        nrzgauss = phase_to_nrz(phase,1);
    else
        nrzgauss = phase_to_nrz_filter(phase,fs);
    end
end
```

```
function [Ibb,Qbb] = mod_sig_downconv(mod_sig,chan,fs,offset)

n  = length(mod_sig);

% Time vector for the length of mod_sig.
t  = 1/fs:1/fs:n/fs;

if (nargin < 4) || (offset == 0)
    fc = 2e6 + (chan * 1e6);
else
    fc = 2e6 + (chan * 1e6) + offset;
end

% Local oscillator signals
LO_I = cos(2*pi*fc*t)';
LO_Q = sin(2*pi*fc*t)';

% In-Phase baseband signal
Ibb = mod_sig.*LO_I;
% Quadrature baseband signal
Qbb = mod_sig.*LO_Q;


function [IbbFilt,QbbFilt] = pd_lpf(Ibb,Qbb,fs,type)

% Pass band freq (normalised to fs/2).
Wp = 2*1e6/fs;
% Stop band freq (normalised to fs/2).
Ws = 2*6e6/fs;

if(nargin<4)||(type==0)
    % 1dB ripple in passband and 80dB atten at Ws.
    [n,Wn] = buttord(Wp,Ws,1,80);
    % Butterworth IIR filter coefficients.
    [b,a] = butter(n,Wn);
else
    % 1dB ripple in passband and 80dB atten at Ws.
    [n,Wn] = cheb2ord(Wp,Ws,1,80);
    % Chebyshev Type II IIR filter coefficients.
    [b,a] = cheby2(n,80,Wn);
end

% Low pass filter baseband signals
IbbFilt = filter(b,a,Ibb);
QbbFilt = filter(b,a,Qbb);
```

```
function phase = phase_extract(IbbFilt,QbbFilt)

w_phase = atan2(QbbFilt,IbbFilt);

% Unwrap +-2pi phase discontinuities from
% arctangent calculation.
phase = unwrap(w_phase);


function nrz_gauss = phase_to_nrz_filter(phase_sig,fs,order)

% Default filter order.
default_order = 1001;

if nargin < 3
    order = default_order;
else
   if order/2 == floor(order/2)
       error('Use odd filter order.');
   elseif order == 1
       order = default;
   end
end

% Design filter.
b=remez(order,[0 1],[0 pi*fs],'d');

% Differenciate the phase signal.
nrz_gauss = filter(b,1,phase_sig);
```

### A.2.3 Slicing Detector

After demodulation the received data is recovered with a simple slicing detector as described in section 4.2.2. The input argument `offset` is the constant number of samples from the beginning of the input data, `nrz`, to the centre of the first symbol of the preamble. These additional samples are produced by the various filtering processes in the demodulation and can be ignored.

```
function msg = int_and_dump_1(nrz,fs,offset);

samp_per_sym = fs*1e-6;
n = length(nrz);
```

```
symbols = floor((n-offset)/samp_per_sym);
for i = 0:symbols
    if nrz((offset + (i*samp_per_sym))) > 0
        msg(i+1) = 1;
    else
        msg(i+1) = -1;
    end
end
```

## A.3 Adaptive Threshold IaD Decision Algorithm Code

The adaptive threshold IaD decision algorithm is implemented in two stages as described in section 7.2. First the new threshold value is calculated by `get_thold`. The variable `type` controls whether all the preamble samples or only the central samples are used in the calculation. Secondly the received data is recovered by a bit slicer, `int_and_dump_adaptive`, that has been modified to take into account the new threshold value. Although these functions are run sequentially in the simulation, the new threshold is not applied until the fifth symbol, that is after the preamble. In both functions `offset` has the same meaning as in `int_and_dump_1` above.

```
function thold = get_thold(nrz,offset,fs,type);

thold = 0;
samp_per_sym = fs*1e-6;

if type == 0 % All samples calculation.
    % samp_per_sym could be odd.
    start = offset - (floor(samp_per_sym/2));
    finish = start + (4*samp_per_sym);
    for i=start:finish
        thold = thold + nrz(i);
    end
    thold = thold/(finish-start);
else % Central samples only.
    for i=0:3
        thold = thold + nrz((offset + (i*samp_per_sym)));
```

```
    end
    thold = thold/4;
end


function msg = int_and_dump_adaptive(nrz,fs,offset,thold);

samp_per_sym = fs*1e-6;
n = length(nrz);

symbols = floor((n-offset)/samp_per_sym);
for i = 0:3
    if nrz((offset + (i*samp_per_sym))) > 0
        msg(i+1) = 1;
    else
        msg(i+1) = -1;
    end
end

for i = 4:symbols
    if nrz((offset + (i*samp_per_sym))) > thold
        msg(i+1) = 1;
    else
        msg(i+1) = -1;
    end
end
```

# Appendix B

# Publications and Presentations

The following publications and invited presentations arose directly from the work described in this thesis.

## B.1 Publications

1. Craig Robinson and Alan Purvis, "A VXIBus Based Bluetooth Radio Test Set for IC Validation & Debug", in *Proceedings of the 8th IEEE International Mixed Signal Testing Workshop*, Montreux, Switzerland, 18-21 June 2002, pp 89-96.

2. Craig Robinson, Alan Purvis, Armin Lechner and Michael Hoy, "Characterisation of Bluetooth Carrier Frequency Errors", in *Proceedings of the 9th IEEE International Mixed Signal Testing Workshop*, Seville, Spain, 25-27 June 2003, pp 119-124.

3. Craig Robinson and Alan Purvis, "Demodulation of Bluetooth GFSK Signals under Carrier Frequency Error Conditions", in *Proceedings of the IEE Colloquium on DSP Enabled Radio*, Livingston, Scotland, 22-23 September 2003.

## B.2 Presentations

1. "A VXIBus Based Bluetooth Test System", Test Engineering Summer School, Institute for Systems Level Integration, Livingston, Scotland, August 2002.

2. "Detecting Bluetooth Carrier Frequency Errors" with Armin Lechner, Advantest (Europe) GmbH Open House Event, Munich, Germany, April 2003.